

MySQL Datenbankhandbuch

Guido Stepken (stepken@little-idiot.de)

Version 0.99 beta, 15. September 1999

Dieses Handbuch behandelt alle Dinge, die man/frau zum Aufbau von Datenbanken im Intranet, Internet oder auch zur Synchronisation von Datenbanken wissen muß. Es sind darüberhinaus praxisorientierte Tutorials für Einsteiger enthalten, die Themen, wie Normalisierung, Erstellung von ER - Diagrammen ohne theoretischen Ballast behandeln. Ziel dieses Handbuches ist es, den Einsteiger so schnell wie möglich in die Lage zuversetzen, mit der kostenlosen MySQL Datenbank und Microsoft Software (ACCESS, EXCEL) als Frontend, auf einfachem Wege fertige Lösungen zu produzieren. Hierzu sind in diesem Handbuch fertige Projekte enthalten, die die Einrichtung von Datenbanken mit den zugehörigen Interfaces von A-Z detailliert beschreiben. Es sind natürlich auch alle Quellcodes genauestens dokumentiert. Dies versetzt den absoluten Einsteiger in die Lage, anhand der Beispiele viel zu lernen, und den Fortgeschrittenen in die Lage, innerhalb von wenigen Minuten diese Beispiele auf seiner Plattform fertig zu installieren, zu testen und zu erweitern. Wir versetzen Sie in die Lage, innerhalb von ***einer Minute*** *eine komplette SQL Datenbank zu* ***installieren und zu starten !*** *Das geht nicht ? Alles geht, nur der Frosch, der muß hüpfen !* ***Hier der Beweis:*** *Live Demo !. Dieses Handbuch enthält darüber hinaus das originale MySQL Handbuch in der aktuellsten Version 3.23. MySQL ist die schnellste SQL Datenbank überhaupt und eignet sich hervorragend für den Einsatz im Internet. Leider behaupten böse Zungen, daß MySQL einige Features ausgewachsener Datenbanken nicht hätte, daher sind im Kapitel Workarounds Lösungen angegeben, die in der Praxis völlig ausreichend sind. Desweiteren enthält dieses Handbuch ausführliche Anleitungen zur Installation von MySQL unter Windows 98/NT und LINUX, sowie Tutorials für MySQL Interfaces in PERL, JAVA und PHP3, und zur Strukturierung und Planung von Datenbanken. Augenmerk wurde auch auf die Möglichkeit gelegt, mit MySQL verteilte Datenbanken zu realisieren, da hier noch einiger Informationsbedarf besteht. Entsprechende PERL Skripte sind natürlich enthalten. Es gibt für MySQL hervorragende Frontends, mit welchen das Anlegen von Datenbanken, das Einrichten von Usern und der Import von Datensätzen aus anderen Datenbanken ein Kinderspiel ist. Diese werden hier ausführlich in Form von Tutorials vorgestellt. Für Microsoft Anwender, die MySQL als Datenbank im Internet und Intranet nutzen möchten, sind Tutorials zu Anbindung von ACCESS, EXCEL und Winword über die ODBC Schnittstelle enthalten.*

1. Download des MySQL Handbuches

- 1.1 Mirrors zum Laden des Handbuches

2. Einführung

- 2.1 Konventionen und Schreibweisen

3. Tutorials für Einsteiger

4. Projekte PHP3, PERL und MySQL

5. Links zu Projekten mit PHP und MySQL

6. Einsteiger Tutorial LINUX MySQL Server

- 6.1 Welches LINUX ?
- 6.2 Start von MySQL
- 6.3 Die MySQL Werkzeuge
- 6.4 Die Standardinstallation unter LINUX
- 6.5 Ändern des Administratorpaßwortes unter MySQL
- 6.6 Remote Administration des MySQL Servers
- 6.7 Vorstellung der Clients für MySQL
- 6.8 Datensatz in die Tabelle einfügen
- 6.9 Exportieren von Datensätzen aus MySQL
- 6.10 Import von Datensätzen in MySQL
- 6.11 Einlesen von Datensätzen über das Netzwerk
- 6.12 Administration über PHP3
- 6.13 MySQL und JDBC Klasse 4
- 6.14 Test von PHP3 und MySQL
- 6.15 Update über ODBC Treiber

7. Die Geschichte von SQL

8. MySQL im praktischen Einsatz - Beispiele

9. Grundwissen zum Entwurf einer Datenbank

10. Was ist MySQL ?

11. Das Handbuch und die Autoren

12. Planung der Datenbankstruktur

- 12.1 Vorüberlegungen
- 12.2 Entity Relationship (ER) - Modell
- 12.3 Begriffe aus der Datenbanktechnik
- 12.4 Ziele bei der Struktur der Datenbank

13. Copyright des MySQL Handbuches

14. Die Historie von MySQL

15. Support für MySQL und Programmierschnittstellen

- 15.1 Die Mailing-Listen Beiträge der letzten 2 Jahre als HTML-Archiv
- 15.2 Erstellen eines BUG-Reports
- 15.3 Kommerzieller Support in Deutschland
- 15.4 **MySQL** Serverhosting
- 15.5 Schulungen MySQL / PHP 4.0
- 15.6 Fragen und Antworten - Mailinglisten

16. Das Jahr 2000 Problem

17. Workarounds für MySQL (Für Umsteiger von anderen SQL Datenbanken)

- 17.1 Foreign Keys
- 17.2 ON DELETE, ON UPDATE
- 17.3 UNION
- 17.4 Locks
- 17.5 Cursors
- 17.6 Views

- 17.7 Subselects
- 17.8 Joins
- 17.9 Transaktionen
- 17.10 Stored Procedures
- 17.11 Trigger
- 17.12 Verschlüsselte Übertragung von Daten
- 17.13 Verteilte Datenbanksysteme und Skalierung
- 17.14 Datenbank-Replikation
- 17.15 Große Datenbanken > 2 GByte

18. Neuigkeiten bei MySQL 3.23

19. Wie stabil ist MySQL ?

20. Unterstütze Betriebssysteme und Bibliotheken

- 20.1 UNIX und UNIX Derivate
- 20.2 MySQL unter Windows 95/98/NT
- 20.3 Mehrprozessor - Systeme
- 20.4 Auswahl von Hardware für MySQL

21. Unterschiede zwischen MySQL-Win32 und UNIX

22. Links und Verweise auf andere Server:

- 22.1 **Nützliche MySQL**-Links
- 22.2 Web-Server mit **MySQL**-Unterstützung
- 22.3 Beispiele zu SQL Datenbanken
- 22.4 Weitere **MySQL**-Beispiele mit Quellcode
- 22.5 Linklisten zu **MySQL**
- 22.6 SQL und Datenbank-Interfaces
- 22.7 ODBC Links

23. Installation

- 23.1 Download
- 23.2 Installations-Verzeichnisse
- 23.3 Kompilieren von **MySQL**

24. Installation der Source-Distribution Schritt für Schritt

- 24.1 Konfigurations-Schalter:
- 24.2 Konvertierung von Zeichensätzen
- 24.3 Einspielen eines Patches
- 24.4 Anmerkungen zum Compilieren
- 24.5 Anmerkungen zu MIT-pthreads - FreeBSD
- 24.6 Anmerkungen zu Perl
- 24.7 Anmerkungen zu bestimmten System/OS-Kombinationen

25. Installation der Binary-Distribution von MySQL

- 25.1 PERL installieren
- 25.2 Starten des Servers
- 25.3 LINUX RPM-Installation
- 25.4 Post-Installation
- 25.5 Probleme beim Start von **MySQL**
- 25.6 Automatischer Start/Stop von **MySQL**
- 25.7 Globale Konfigurationsdateien (Option-Files)

26. Zugriffsberechtigungen (Privilege-System)

- 26.1 Einleitung
- 26.2 Welche Aufgaben werden vom Berechtigungssystem erledigt?
- 26.3 Wie funktioniert das Sicherheitssystem?
- 26.4 user-Tabelle
- 26.5 db-Tabelle
- 26.6 host-Tabelle
- 26.7 Definitionen

- 26.8 Festlegen von Berechtigungen
- 26.9 Weitere Beispiele:
- 26.10 Default-Berechtigungen

27. Zugriffsprobleme

- 27.1 Checkliste bei Zugriffsproblemen

28. MySQL-Sprachreferenz mit vielen Beispielen

- 28.1 CREATE TABLE
- 28.2 OPTIMIZE TABLE SYNTAX
- 28.3 ALTER Table
- 28.4 DROP Table
- 28.5 Typenveränderungen durch den MySQL Optimizer
- 28.6 Reihen - Formate (row format)
- 28.7 DELETE
- 28.8 SELECT
- 28.9 JOIN
- 28.10 INSERT
- 28.11 REPLACE
- 28.12 LOAD DATA INFILE
- 28.13 UPDATE
- 28.14 USE
- 28.15 FLUSH
- 28.16 KILL
- 28.17 SHOW
- 28.18 EXPLAIN (Tuning der Geschwindigkeit)
- 28.19 DESCRIBE
- 28.20 LOCK/UNLOCK von TABELLEN
- 28.21 SET OPTION
- 28.22 GRANT und REVOKE
- 28.23 CREATE INDEX
- 28.24 DROP INDEX
- 28.25 Comment
- 28.26 CREATE FUNCTION/DROP FUNCTION

- 28.27 Ist MySQL empfindlich bei reservierten Worten ?

29. Wie sichere ich MySQL gegen Hacker ab ?

30. ODBC Treiber für MySQL

- 30.1 ODBC-Treibertypen
- 30.2 ODBC Kompatibilitätsstufen
- 30.3 API-Kompatibilitätsstufen
- 30.4 SQL-Kompatibilitätsstufen
- 30.5 Installation des MyODBC-Treibers für ACCESS
- 30.6 MyODBC
- 30.7 Kompatibilität zu Anwendungsprogrammen
- 30.8 Bekannte Probleme mit MyODBC
- 30.9 Felder in dem ODBC Administrator
- 30.10 Wie man den Wert einer AUTO_INCREMENT Spalte in ODBC ausliest
- 30.11 Optionen
- 30.12 Mögliche Einstellungen für bekannte Clients
- 30.13 Bekannte ODBC Probleme und Ursachen
- 30.14 Bekannte Bugs
- 30.15 Installation des MyODBC Treibers unter Windows 98

31. MySQL Tutorial

- 31.1 Verbinden mit dem Server
- 31.2 Eingabe von Abfragen
- 31.3 Anlegen und Benutzen einer Datenbank
- 31.4 Anlegen und Auswählen einer Datenbank
- 31.5 Anlegen einer Tabelle
- 31.6 Laden von Daten in die Tabelle
- 31.7 Abfragen der Informationen aus der Tabelle
- 31.8 Abfrage aller Daten
- 31.9 Auswahl spezieller Einträge
- 31.10 Auswählen von Spalten
- 31.11 Sortieren von Reihen
- 31.12 Berechnung von DATUM

- 31.13 NULL Werte
- 31.14 Abfrage von Suchmustern (Patterns)
- 31.15 Zählen in Spalten
- 31.16 Verknüpfen von Tabellen
- 31.17 Wie man Informationen über Datenbanken und Tabellen erhält
- 31.18 MySQL im Batch Modus

32. Begriffsdefinitionen für Einsteiger

- 32.1 Attribut
- 32.2 Primärschlüssel (primary key)
- 32.3 Fremdschlüssel (foreign key)
- 32.4 INDEX
- 32.5 Relation (relation)
- 32.6 Aggregat Funktionen
- 32.7 Tupel
- 32.8 SQL DML (Data Definition Language)
- 32.9 SQL DDL (Data Definition Language)

33. Installation von ODBC Treibern für Applikationen

- 33.1 ODBC Grundlagen (Open Database Connectivity)
- 33.2 ODBC Treiberlevel
- 33.3 ODBC Entwicklungs-Kits
- 33.4 ODBC mit PERL

34. MySQL und JAVA Frontends

- 34.1 Vor-und Nachteile von JDBC Treibern
- 34.2 Cursor unter JDBC
- 34.3 Bibliotheken für JAVA (JDBC) Frontends
- 34.4 Kompilieren und Start eines Beispiels
- 34.5 Datentypen - Konvertierung MySQL<->JAVA
- 34.6 JAVA Bibliotheken zum Zugriff auf SQL Datenbanken
- 34.7 Links und Beispiele

35. Anbindung von ACCESS an MySQL

36. Tutorial Testverbindung von PHP3 und MySQL

37. Tutorial für PHP 3.0/4.0 WWW-Interfaces

- 37.1 Vorbereitung
- 37.2 Werkzeuge zum Programmieren
- 37.3 Provider für MySQL und PHP
- 37.4 Installation der notwendigen Komponenten

38. Installation von MySQL

- 38.1 Installation von PHP 3.0 unter Windows 98
- 38.2 Installation von PHP3 3.0.x unter LINUX

39. PHP3 3.0 Tutorial Schritt für Schritt

- 39.1 Tutorial PHP 3
- 39.2 Grundsätzliches zum Tutorial
- 39.3 Ein erster Test von PHP 3.0/4.0
- 39.4 Sonderzeichen
- 39.5 Variablen
- 39.6 Vordefinierte Variablen mit spezielle Funktionen
- 39.7 Einfache Rechenoperationen
- 39.8 Schleifen
- 39.9 Weitere Befehle zur Ablaufkontrolle
- 39.10 Fallunterscheidungen
- 39.11 Vergleiche
- 39.12 Wie verwende ich Funktionen und Bibliotheken ?
- 39.13 Vordefinierte Variablen
- 39.14 Dateien einbinden
- 39.15 Dateien lesen und schreiben
- 39.16 Bildbearbeitung
- 39.17 Zugriff auf Datenbanken
- 39.18 SQL-Befehle mit PHP 3

- 39.19 Weitere SQL-Befehle
- 39.20 Funktionen
- 39.21 String - Funktionen
- 39.22 Cookies - Mechanismen
- 39.23 Arrays aus Formularen
- 39.24 Ein paar Beispiele zu MySQL und PHP3
- 39.25 Systemnahe Funktionen: Arbeiten mit Sockets
- 39.26 Was für ein WWW-Server läuft auf dieser Domain ?
- 39.27 Überprüfung von erlaubten Buchstaben mit EREG()
- 39.28 Upload einer Datei

40. Projekt Einfacher Datenbankzugriff mit MySQL und PHP

- 40.1 Schema des Datenbankzugriffs
- 40.2 Der erste Kontakt
- 40.3 Auswahl der Datenbank
- 40.4 Anfrage an MySQL
- 40.5 Schließen der Verbindung
- 40.6 Übersicht der verwendeten Funktionen
- 40.7 Ein größeres Beispiel
- 40.8 Tips und Tricks
- 40.9 Fehlerkontrolle

41. Projekt Adressen - Datenbank

42. Nützliche PHP 3.0/4.0 Bibliotheken

43. Erstellung der ER-Diagramme

- 43.1 Darstellung der Entitätstypen und Relationen
- 43.2 Wertigkeiten der Relationen
- 43.3 Wie entwickelt man denn nun ein solches ER-Diagramm ?
- 43.4 Systematische Entwicklung anhand eines Beispiels
- 43.5 Was kann man aus diesem Diagramm nun entnehmen ?

44. Tutorial Normalisierung von Datenbanken

- 44.1 Sinn und Zweck der Normalisierung
- 44.2 Obsthändler Krämers erster Datenbankentwurf
- 44.3 Definition der Normalformen
- 44.4 Anomalien
- 44.5 Praxisanwendung Normalisierung

45. Lexikon der Datenbanktechnik

46. Verteilte Datenbanken

- 46.1 Datenbank Spiegelung im Intranet
- 46.2 Lastverteilung von SQL-Datenbanken im Intranet
- 46.3 Lastverteilung von SQL-Datenbanken im Internet
- 46.4 Verteilte Datenbanken mit Standleitung
- 46.5 Verteilte Datenbanken mit DIAL-ON-DEMAND Leitung
- 46.6 Verteilte Datenbanken mit Mail-Synchronisation
- 46.7 PERL Dämon zur Synchronisation dezentraler Datenbanken
- 46.8 Replikation von MySQL Datenbanken mit PERL in Echtzeit
- 46.9 Verschlüsselter Datentransfer
- 46.10 Verschlüsselte Festplatten

47. SQL Datenbank Beispiel Inventur aus der Zeitschrift i'X

48. Bild - Datenbank und Bildermanipulation mit ImageMagick

49. Tutorial MySQL Adressen - Datenbank

- 49.1 Datenbank ohne Datenbank
- 49.2 Suchen und Finden

50. Projekt MySQL und PHP3 für Fortgeschrittene

51. MySQL Performance Benchmarks

52. Alternative SQL Datenbanken

- 52.1 Live Tutorial !
- 52.2 HyperSQL für Einsteiger
- 52.3 Streamstore Datenbank
- 52.4 Instantdb

53. MySQL unter Microsoft Windows 98/NT

- 53.1 Installation von MySQL unter Windows 98/NT
- 53.2 Die MySQL Binaries - kurz erklärt !
- 53.3 Windows und UNIX/NT kombiniert
- 53.4 MySqlManager

54. MySQLWinAdmin

- 54.1 Der phpMyAdmin Manager

55. MySQL und ACCESS - Vorüberlegungen

- 55.1 Überlegungen zur Performance
- 55.2 MySQL - Performancevergleich
- 55.3 Performance der ODBC-Treiber
- 55.4 Migration zu Microsoft SQL
- 55.5 Typische Probleme beim Konvertieren der ACCESS Datenbank
- 55.6 Variablentypen unter ACCESS und MySQL
- 55.7 Verknüpfte Tabellen mit MySQL
- 55.8 Eingabemasken von ACCESS 97
- 55.9 Replikationen von Datenbanken unter ACCESS

56. Frontends und Cursor unter MySQL

- 56.1 Seitenweise Ausgabe von Daten
- 56.2 Cursor mit PHP3
- 56.3 Cursor mit PERL
- 56.4 Funktionierende Cursor mit der C API unter MySQL
- 56.5 Seitenweise Ausgabe mit ASCII Datenbanken
- 56.6 Cursor mit ACCESS und MySQL
- 56.7 Cursor mit Visual C++/BASIC und MySQL
- 56.8 Frontend mit ODBC
- 56.9 Cursor Bibliotheken mit Python
- 56.10 Cursor mit JDBC
- 56.11 Beispiele und ausgewählte Links !!!!!

57. Ausgewählte Links

- 57.1 ODBC Links
- 57.2 JDBC Programmierung und Treiber
- 57.3 Datenbank Ansteuerung unter PERL
- 57.4 MySQL Tutorials

58. Projekt Kontakt-Datenbank

- 58.1 Beschreibung
- 58.2 Das WWW-Interface
- 58.3 Tabelle 1a
- 58.4 Tabelle 1b
- 58.5 Tabelle 1c
- 58.6 Tabelle 1d
- 58.7 PHP Listing

59. LAMP (Linux Apache MySQL PHP) und CHROOT()

- 59.1 Apache WWW-Server Installation unter /home/www/
- 59.2 Nachträgliche Installation von PHP 3 (beta)

- 59.3 Apache und PHP 3

60. Danksagung






1. Download des MySQL Handbuches

Das MySQL Handbuch ist zwar noch sehr BETA, aber für viele schon das Referenzhandbuch in deutscher Sprache. Es ist komplett in SGML geschrieben, also automatisch auch in PDF, RTF, HTML, LYX, TEX, u.s.w. verfügbar. Die PDF Version ist viel zu groß und auch momentan in Überarbeitung für ein Buch mit ca. 600 Seiten, welches in ca. 14 Tagen in Druck geht. Vormerkungen für dieses MySQL - Buch bitte an:

vormerkung-mysql@little-idiot.de.

Momentan ist also nur die gezippte Version des Handbuches verfügbar. Ich möchte auch alle Leser bitten, diese Versionen von einem zufällig ausgewählten Mirror zu laden, da die Serverlast doch erheblich sein dürfte. Die Live-Demo ist natürlich nicht enthalten. Nachdem das JAVA Applet im Browser gestartet ist, läuft die Demo auch offline.

Viel Spaß beim Ausprobieren der Beispiele !







52.1 Live Tutorial !

Sie glauben auch noch nicht daran, daß man innerhalb einer **einzigen Minute** eine komplette SQL Datenbank aus dem Internet downloaden, auf der Arbeitsstation installieren und starten kann ? Dann aktivieren Sie bitte die JAVA Unterstützung in Ihrem Browser, und klicken Sie **http://www.rent-a-database.de/mysql/mysqllive.html**. Viel Spaß nun beim Ausprobieren der Beispiele aus den MySQL Tutorials !

Dieses Beispiel hier zeigt schön, daß man ohne Installation von Software auf den Clients mit Datenbanken komfortabel arbeiten kann. Was das für die Zukunft der Systemadministratoren bedeutet, kann sich wohl jeder ausmalen....Mit Applixware Office, dem inzwischen kostenlosen Staroffice 5.1 (gerade von SUN übernommen und für kommerzielle Nutzung freigegeben) und CorelOffice dürfte die Zeit der aufwendigen Installation von Softwarepaketen vorbei sein. Hier wurde ein Klasse 4 JDBC Interface verwendet. Beachten Sie auch, daß Sie mit der Datenbank auch arbeiten können, ohne daß die Internet-Verbindung besteht ! - Zauberei ?







17. Workarounds für MySQL (Für Umsteiger von anderen SQL Datenbanken)

Wer sich bereits mit anderen SQL Datenbanken auskennt, der wird sicher bei MySQL einige Funktionen vermissen, die in anderen Datenbanken selbstverständlich sind. Für MySQL sind in diesem Kapitel viele Workarounds beschrieben, die diese Funktionen zumeist vollständig ersetzen. Da bei MySQL viele Pre-und Postprozessoren einfach nicht existieren, kann MySQL erheblich schneller sein, als alle teuren, professionellen Datenbanken. Für Internet-Anwendungen mit vielen simultanen Clients können diese Pre - und Postprozessoren die Performance erheblich verlangsamen. Unter MySQL werden diese Performancekiller erst garnicht unterstützt. MySQL ist als Internet-Datenbank mit vielen tausend simultanen Clients konzipiert. Zur Beschleinigung von ACCESS 95/97/2000 ist MySQL ohne Probleme einsetzbar, da ACCESS als Frontend fast alle der hier aufgeführten Features entweder nicht unterstützt (Subselects) oder auf die eine oder andere Art bereits implementiert hat (clientside cursor)... Nun aber zu den Workarounds:

1. Foreign Keys
2. ON DELETE, ON UPDATE
3. Union
4. Locking
5. Cursors
6. Views
7. Transaktionen (COMMIT/ROLLBACK)
8. Subselects (mehrfache SELECT hintereinander)
9. Joins (right join, ...)
10. Verschlüsselte Übertragung von Daten
11. Stored Procedures
12. Trigger
13. Verteilte Datenbanksysteme und Skalierung
14. Datenbank-Replikation
15. Große Datenbanken > 2 Gigabyte







1.1 Mirrors zum Laden des Handbuches

Vielen Dank allen, die spontan Ihre Hilfe angeboten haben. Weitere Mirrors werden noch gesucht.....

Vielen Dank schon einmal an die Betreiber der Server von:

http://www.php-center.de, http://www.xsl.de, http://www.php-homepage.de für die freundliche Unterstützung.

Nun zu den ZIP Dateien:

http://www.php-center.de/mirrors/mysql/mysql.zip

http://www.xsl.de/mysql/mysql.zip

http://www.php-homepage.de/mysql/mysql.zip

http://www.rent-a-database.de/mysql/mysql.zip







2. Einführung

Dieses Handbuch beruht auf den Dokumentationen der gerade aktuellen **MySQL** Distribution 3.23. Insbesondere wurde viel Wert auf die genaue Übersetzung der MySQL - Sprachreferenz gelegt, die wichtige Details der Befehlsyntax von MySQL enthält. Darüber hinaus werden aber auch alle Probleme angesprochen, die insbesondere Einsteigern große Schwierigkeiten bereiten. Tutorials sollen auch Einsteiger in die Lage versetzen, MySQL auf Ihrer Arbeitsstation zu installieren und Anwendungsprogramme, wie z.B. Microsoft Excel, Access, Foxpro, DBase und Winword über ODBC Schnittstellen an MySQL anzubinden. Darüberhinaus werden solide Grundkenntnisse in der sehr einfachen Programmiersprache PHP, SQL und auch PERL vermittelt. Selbstverständlich sind viele Beispiele enthalten, die dem Anfänger das **AHA - Erlebnis** und viele Zusammenhänge auch vermitteln sollen. Da das originale Handbuch Einsteigern wenig über die Programmierspache SQL vermittelt, wurde natürlich ein ausführliches Tutorial zu der SQL Programmiersprache hinzugefügt.

Viele Beispiele in diesem Tutorial sind natürlich auf unseren Datenbankserver im Internet abgestimmt, der so installiert wurde, daß die hier genannten Beispiele ohne große Probleme funktionieren.

Unser Anliegen ist es, Ihnen auf einfachstem Wege moderne Datenbanktechnik und das sogenannte PACK AN zu vermitteln. Alle in diesem Buch angeführten Beispiele sind auf unser **Rent - A - Database** Angebot zugeschnitten. Daher könnte es passieren, daß bei anderen Providern einige Beispiele nicht direkt funktionieren. Wir weisen in unserem Handbuch aber auf eventuelle Probleme hin, sodaß Sie auch bei der Installation eines Intranet-Servers mit MySQL keine Probleme haben sollten.

Sie werden beim Einsatz von MySQL feststellen, daß diese Datenbank, obwohl sie der GPL (GNU PUBLIC LICENSE) unterliegt, also ohne Lizenzgebühren genutzt werden darf, keine minderwertige Software ist, sondern diese einige Features besitzt, die Sie verzweifelt bei anderen Datenbanken vermissen werden, wie z.B. das Privilegien-System. Außerdem wird diese ständig weiterentwickelt. Support von den Programmierern direkt ist ebenfalls erhältlich.

Ich möchte daher alle Leser dieses Handbuches dazu ermutigen die sogenannten Lizenzgebühren an die Entwickler von MySQL zu entrichten ..... Diese haben viel Arbeit in MySQL und in die Installationsroutinen gesteckt - Sie werden sehen - ein Knopfdruck - und MySQL ist auf Ihrer Arbeitsstation installiert. Das spart Ihre Zeit, viel Recherche und macht sich für Sie bezahlt. Natürlich gibt es auch eine GPL Version von MySQL, die allerdings einige hier in diesem Handbuch erwähnten, äußerst nützlichen Features noch nicht besitzt, wie z.B. die Möglichkeit, von Ihrer Arbeitsstation oder von Ihrer SAGE KHK Datenbank die Daten in Ihren Online-Shop, der unter MINIVEND, LITTLE-IDIOT SHOP oder INTERSHOP läuft, zu übertragen. INTERSHOP z.B. ist ebenso, wie MINIVEND (was inzwischen nicht mehr so MINI ist, in PERL programmiert. (Kein Scherz, ich habe es hier unter LINUX laufen ...). Sie finden die Kontonummern der Postbank in Stockholm hier http://www.tcx.se

Wir wünschen nun viel Erfolg !







2.1 Konventionen und Schreibweisen

Es werden einige Konventionen verwendet, die einer kleinen Erklärung bedürfen:

```
shell>
```

ist der Prompt des Betriebssystems. Das kann z.B. ein DOS oder UNIX Prompt sein. Unter UNIX bzw. LINUX ist dies meist die **CSH** oder die **BASH**. Unter **DOS/Windows** ist dies der **command** prompt des DOS - Fensters

```
mysql>
```

Hier ist bereits die Verbindung zu MySQL mit Hilfe des MySQL Clients **mysql** hergestellt und MySQL erwartet Statements.

```
mysql> DROP TABLE [IF EXISTS] tblname
```

zeigt an, daß der Ausdruck in eckigen Klammern optional ist.







3. Tutorials für Einsteiger

Hier eine kurze Liste von kleinen und größeren Tutorials oder kleinen, lehrreichen Abschnitten in diesem Handbuch. Diese Tutorials umfassen alles, was man über Datenbanken wissen muß, sind jedoch von Einsteigern für Einsteiger geschrieben:

- Live Tutorial SQL ! Reinschauen !
- Normalisierung von Datenbanken
- Einsteiger Tutorial LINUX MySQL Server
- Kleine und größere, typische Probleme beim Verständnis von JOINS
- Wie finde ich heraus, warum manche Befehle so langsam sind ?
- MySQL Tutorial zum Erlernen der Sprache SQL
- PHP3 Tutorial Grundlagen der Sprache PHP
- Tutorial Testverbindung von PHP3 und MySQL







4. Projekte PHP3, PERL und MySQL

In diesem Kapitel finden Sie immer neue Projekte, die den Einsatz von PHP3 und MySQL erfordern. Im Gegensatz zu unseren Tutorials sind Projekte von A-Z bis ins Detail genau beschrieben. Es sind alle Erklärungen zu wichtigen Code - Abschnitten enthalten sowie die **CREATE** Statements für Ihren MySQL Datenbankserver. Die Inbetriebnahme unserer Projekte sollte im Allgemeinen nicht mehr als 10 Minuten in Anspruch nehmen, sofern Ihr WWW-Server mit PHP3, allen erforderlichen Libraries und einer MySQL Datenbank ausgestattet ist. Die Beispiele sind alle genauestens auf unserem Datenbankserver getestet worden. Auf eventuell erforderliche Änderungen auf Ihrem Datenbankserver weisen wir im Text hin. (Ohne Garantie)

- Projekt Adressen - Datenbank
- Projekt Einfacher MySQL Zugriff mit PHP3
- Projekt Bild - Datenbank und Manipulation mit Imagemagic
- Projekt Adressendatanbank mit PERL
- Projekt MySQL und PHP3 für Fortgeschrittene







5. Links zu Projekten mit PHP und MySQL

In diesem Kapitel werden nette und gut dokumentierte Projekte vorgestellt, die nicht Teil dieses Handbuches sind, die man sich jedoch einmal Anschauen sollte.

- Mark Kronsbein beschreibt auf seiner Homepage: http://www.php-homepage.de/ die Auswertung einiger Browser-Variablen mit PHP3 und den Anschluß an MySQL. Somit finden sich Browsertyp und die URL, die der Surfer zuvor besucht hat, direkt in der MySQL Datenbank zur Weiterverarbeitung wieder. Das Werkzeug ist interessant, insbesondere wenn man feststellen möchte, welche Werbeaktion im WWW die erfolgreichste war. Allein die Auswertung von Serverstatistiken reicht da wohl nicht mehr aus.
- Auf der Website von http://www.caupo.de findet sich ein Open Source SHOP, der auch frei genutzt werden darf. Der SHOP besitzt eine kleine Kundenverwaltung, sodaß ein Kunde nicht alle Daten bei einer Bestellung neu eingeben muß. Dies ist heutzutage die Minimalanforderung an einen SHOP. Der Code ist frei verfügbar, gut dokumentiert und es gibt auch Support in einer Mailing-Liste.
- Uwe Derbacher beschreibt ausführlich die Kombination HTML-Formulare mit PHP/FI, PERL und mSQL. Das Beispiel läßt sich einfach auf MySQL übertragen: http://www.debacher.de/php/php_msql.htm. Dasselbe Beispiel in PERL realisiert findet sich hier: http://www.debacher.de/perl_msql/welcome.htm. Hier kann man gut PHP mit PERL vergleichen.....
- Ein kommerzieller SHOP unter PHP3 mit MySQL als Basis: http://www.wse.de/shop/
- Eine freier SHOP unter PHP mit MySQL als Basis: http://www.camelcity.com
- Ein Internet Marktplatz von Tobis Tatschiller in der Zeitung Internet-Professionell, 02/99: Internet Marktplatz







6. Einsteiger Tutorial LINUX MySQL Server

Dieses Tutorial beschreibt ein Projekt zum Aufbau eines MySQL Servers unter LINUX oder UNIX allgemein für die Nutzung im Internet oder im Intranet. Die Beispiele sind aber auch auf eine Installation von MySQL unter Microsoft Windows übertragbar.

Wer sich genau an dieses Tutorial hält, der wird recht schnell das erste Erfolgserlebnis mit MySQL haben - auch als völliger Einsteiger ! Es soll als sogenanntes **PACK AN** dienen, um eventuelle Ängste bei dem ersten Kontakt mit der unbekannten Materie SQL Datenbanken und vor allem mit der Bedienungsoberfläche abzubauen.

Besonderen Wert wurde auf die Vergabe der Zugriffrechte unter MySQL und auf die Bedienung über eine Windows Arbeitsstation gelegt. Hierzu werden auch alle wichtigen Werkzeuge zur Bedienung von MySQL unter Windows vorgestellt.

Vielleicht noch ein Hinweis auf das Tutorial Tutorial Normalisierung von Datenbanken. Bevor Sie Datenbankstrukturen anlegen, sollten die typischen Fehler von **"Obsthändler Krämer"** einmal genau nachvollziehen. Danach werden Ihnen einige seltsamen Dinge, die bei Datenbanken auftreten können (Stichwort Anomalien) erheblich karer werden.

Viel Erfolg !







6.1 Welches LINUX ?

Inzwischen werden alle LINUX Distributionen mit deutschem Installationshandbuch ausgeliefert. Es ist hierbei fast egal, welche Distribution Sie einsetzen, da es in fast allen Fällen erforderlich ist, MySQL neu zu installieren und zu konfigurieren.

Es ist dabei auch völlig egal, wie alt die LINUX Distribution ist, da alle Kernel Versionen ab 2.0.34 aufwärts absolut stabil laufen. Dieses Handbuch enthält eine Anleitung, wie Sie sich stets die aktuellste MySQL Version auf Ihrem Server installieren können. Wir gehen im folgenden nun davon aus, daß Ihre Distribution standardmäßig mindestens MySQL 3.21 installiert hat. Alle Beispiele hier wurden auf einer S.u.S.E. Distribution 6.0 erstellt, die gegenüber anderen Distributionen, wie RedHat oder Debian ein paar kleine Nachteile hat, die in diesem Tutorial jedoch angesprochen werden. Für Einsteiger, die noch keine Erfahrung mit LINUX gemacht haben, seien die CALDERA, SUSE oder RedHat Distributionen empfohlen. DEBIAN LINUX bietet einige Vorteile, die jedoch nur von Profis geschätzt werden.




6.2 Start von MySQL

Um MySQL zu starten, muß man sich bei LINUX unnötigerweise als **root** einloggen, da die S.u.S.E. Startup Skripte MySQL unnötigerweise als ROOT starten möchten. Dies stellt natürlich ein gravierendes Sicherheitsproblem dar, welches im Firewall Handbuch genauestens erläutert ist. Falls Sie gedenken, einen eigenen MySQL Server im Internet zu betreiben, sollten Sie dringend diese Kapitel durchlesen. In einem späteren Kapitel wird genau beschrieben, wie man MySQL korrekt nur mit User-Rechten startet, ohne daß die Funktionalität beeinträchtigt wird.

Falls Sie sich jedoch mit LINUX und MySQL nicht auskennen und beabsichtigen, einfach nur einen MySQL Server im Internet zu nutzen, schauen Sie sich doch einfach unser Angebot an: http://www.rent-a-database.de.

```
bash-2.02$ pwd
/platte2/home/user01
bash-2.02$ su
Password:
bash-2.02# whoami
root
```

MySQL sollte niemals mit ROOT Rechten gestartet werden, einfache User Rechte reichen völlig aus. Wir beschreiben nun trotzdem den Start des MySQL Binary unter S.u.S.E. LINUX 6.0 ohne sicherheitsrechnische Vorkehrungen:

Wir starten nun mit **/sbin/init.d/mysql start** den Server:

```
bash-2.02# /sbin/init.d/mysql start
Starting service MySQL.                                  done
bash-2.02#
```

Der Server ist nun gestartet, leider haben wir keine Informationen über Versionsnummer, Portnummer, Verzeichnisse u.s.w. Wie findet man das heraus ?

Hierzu verwenden wird einige elementare Netzwerk Befehle unter UNIX:

```
bash-2.02# ifconfig
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Bcast:127.255.255.255  Mask:255.0.0.0
          UP BROADCAST LOOPBACK RUNNING  MTU:3584  Metric:1
          RX packets:177 errors:0 dropped:0 overruns:0 frame:0
          TX packets:177 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0

eth0      Link encap:Ethernet  HWaddr 00:80:AD:30:B6:CA
          inet addr:10.0.0.5  Bcast:10.0.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:195 errors:0 dropped:0 overruns:0 frame:0
          TX packets:36 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0
          Interrupt:10 Base address:0x6600
```

ifconfig zeigt uns an, mit welcher IP-Nummer unser MySQL Server im Netz erreichbar ist. Schließlich möchten wir die SQL Datenbank ja allen Usern im Netz unter ACCESS, Winword und EXCEL zur Verfügung stellen.

```
bash-2.02# netstat -an|more
Active Internet connections (including servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:3306            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:6000            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:98              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:40001           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:20005           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:6711            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:113             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:79              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:110             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:513             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:514             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:515             0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:21              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:37              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:19              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:13              0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:9               0.0.0.0:*               LISTEN
--More--
```

Unter S.u.S.E. LINUX sind alle möglichen unnötigen Programme gestartet, die sich an irgendwelche Ports und Netzwerkkarten bunden sind, und somit ein Sicherheitsrisiko darstellen. Uns interessieren im Moment nur die Ports 3333 oder 3306. Die ältere Version von MySQL benutzt Port 3306, die neuere den Port 3333. In der ersten Zeile schon kann man ablesen, daß MySQL den Port 3306 verwendet...

Dieser SQL Server ist also von außen unter der IP-Nummer 10.0.0.5 und der Portnummer 3306 zu erreichen. Das sollten Sie sich irgendwo notieren. Falls Sie unser **Rent - A - Database** Angebot nutzen, schauen Sie bitte in die persönliche Kundeninformation hinein. Dort finden Sie alle Angaben über IP-Nummer und Ports Ihres Datenbank - Servers.

S.u.S.E. Linux hat einige Eigenheiten bei der Grundkonfiguration, die einige Dinge verlangsamen können. Ohne diese Einträge reagiert LINUX beim Einloggen über das Netzwerk mit TELNET z.B. nur sehr verzögert.

Der Grund liegt darin, daß in der Datei **/etc/inetd.conf** leider einige Dienste, wie **echo, chargen, discard** deaktiviert wurden. Mit dem Editor **joe** (Hilfe mit STRG-k h, speichern mit STRG-k x) sollten die Einträge nach einer Korrektur so aussehen:

```
bash-2.02# more /etc/inetd.conf
# See "man 8 inetd" for more information.
#
# If you make changes to this file, either reboot your machine or send the
# inetd a HUP signal:
# Do a "ps x" as root and look up the pid of inetd. Then do a
# "kill -HUP <pid of inetd>".
# The inetd will re-read this file whenever it gets that signal.
#
# <service_name> <sock_type> <proto> <flags> <user> <server_path> <args>
#
echo    stream  tcp     nowait  root    internal
echo    dgram   udp     wait    root    internal
discard stream  tcp     nowait  root    internal
discard dgram   udp     wait    root    internal
```

```
daytime stream  tcp     nowait  root    internal
daytime dgram   udp     wait    root    internal
chargen stream  tcp     nowait  root    internal
chargen dgram   udp     wait    root    internal
time    stream  tcp     nowait  root    internal
time    dgram   udp     wait    root    internal
#
```

Mit dem Kommando: **killall inetd; inetd** starten Sie den INETD Server neu. Die Probleme sollten nun nicht mehr auftreten.

Die Datei /etc/hosts sollte alle IP-Nummern enthalten, mit denen Sie aus dem LAN auf den SQL Server zugreifen möchten:

```
bash-2.02# more /etc/hosts
#
# hosts         This file describes a number of hostname-to-address
#               mappings for the TCP/IP subsystem.  It is mostly
#               used at boot time, when no name servers are running.
#               On small systems, this file can be used instead of a
#               "named" name server.
# Syntax:
#
# IP-Address  Full-Qualified-Hostname  Short-Hostname
#
#10.0.0.9       www2.intra.net          www2
#10.0.0.9       www.intra.net           www
10.0.0.5        tunix.intra.net         tunix
10.0.0.1        client1.intra.net       tester1
127.0.0.1       localhost
bash-2.02#
```

Ein bessere Alternative ist das Aufsetzen des DNS-Servers. Was für Einsteiger bisher noch ein Graus war, läßt sich mit diesem Toolkit einfach erledigen. Hierzu können Sie das Little-Idiot Toolkit verwenden. Die Einträge in den DNS Server wenden dann automatisch auch in die Datei **/etc/hosts** geschrieben.






6.3 Die MySQL Werkzeuge

Zusammen mit MySQL unter LINUX werden eine ganze Reihe von Werkzeugen mitgeliefert.

Tippen Sie hierzu bitte **mysql** in der Shell ein, und drücken Sie dann zweimal auf die **TAB** - Taste. Sie sehen nun einige Befehle, die mit dem Wort **mysql** beginnen:

```
bash-2.02$ mysql
mysql                          mysqlaccess
mysql.server                   mysqladmin
mysql_fix_privilege_tables     mysqlbug
mysql_install_db               mysqldump
mysql_setpermission            mysqlimport
mysql_zap                      mysqlshow
bash-2.02$ mysql
```

Falls Sie diese nicht sehen, dann ist eventuell der Pfad auf das Verzeichnis **/usr/local/bin/** nicht korrekt gesetzt. Sie müssen dann per Hand in dieses Verzeichnis wechseln.

Hier nun die Erklärungen zu den Befehlen:

- **mysql** ist der normale MySQL Client, mit welchem Sie im Prinzip alle Administrationstätigkeiten durchführen können.
- **mysql.server** ist das Start/Stop Skript, mit welchem man den Server starten kann.
- **mysql_fix_privilege_tables** sollten Sie stets dafür verwenden, um direkt zu Beginn die Zugriffsrechte auf MySQL genauer zu definieren.
- **mysql_install_db** ist ein Skript, welches direkt zu Beginn die MySQL Datenbank installiert und schon einmal ein Paar GRANT Tables (dem Rechtesystem) anlegt.
- **mysql_set_permission** ist ein PERL Skript, mit welchem man auf einfache Art und Weise menügesteuert Rechte in der MySQL Datenbank setzen kann.
- **mysqlshow** zeigt alle bereits installierten Datenbanken des MySQL - Servers an.
- **mysql_access** ist das komfortableste Werkzeug, um Rechte in MySQL zu vergeben. Es ist ein sehr umfangreiches PERL Skript.
- **mysqladmin** ist das Werkzeug, um im Batch-Modus Befehle an MySQL zu übergeben. Siehe hierzu auch Kapitel Batch-Modus
- **mysqlbug** sollte man aufrufen, wenn man einen BUG-Report an die Programmierer versenden möchte. Es wird beim Aufruf automatisch der Editor EMACS gestartet, und es erscheint ein Formular, welches Sie genau ausfüllen sollten.
- **mysqldump** ist das Werkzeug, um Daten aus der Datenbank oder auch einzelnen Tabellen auf die Festplatte zu sichern.
- **mysqlimport** ist Ihr Werkzeug, um ASCII Daten in eine Tabelle einzulesen. Hierbei werden viele Optionen unterstützt.

Bei vielen dieser Kommandos kann und muß man Optionen angeben, die sehr nützlich sind, wie z.B. Kompression bei der Ubertragung der Daten in die MySQL Datenbank über das Netzwerk, u.s.w. Rufen Sie hierzu einmal alle Befehle ohne Parameter auf, und sehen Sie sich die Parameter genau an. Dasselbe gilt natürlich auch für die Befehlsoptionen der Windows Version von MySQL.








6.4 Die Standardinstallation unter LINUX

Wir haben nun die MySQL Datenbank gestartet, wissen aber noch nichts über Datenbanken, Tabellen, Rechte u.s.w. Es könnte z.B. sein, daß bereits Datenbanken angelegt sind, oder jeder über das Netzwerk beliebig Daten löschen darf. Daher werden wir nun im Folgenden mit den zuvor vorgestellen Werkzeugen den MySQL Datenbankserver erkunden:

```
user01@tunix:~ > mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4 to server version: 3.21.33b

Type 'help' for help.

mysql> show databases;
+----------+
| Database |
+----------+
| mysql    |
| test     |
+----------+
2 rows in set (0.00 sec)

mysql>
```

Wie wir sehen, sind bereits zwei Datenbanken angelegt worden. Dieselbe Ausgabe liefert übrigens das Shell Werkzeug **mysqlshow**. Nebenbei bemerkt: Man kann Befehle wiederholen, indem man einfach die Taste <Cursor Up> verwendet, das spart viel Tipparbeit.

```
mysql> help

MySQL commands:
help    (\h)    Display this text
?       (\h)    Synonym for Help
clear   (\c)    Clear command
connect (\r)    Reconnect to the server. Optional arguments are db and host
edit    (\e)    Edit command with $EDITOR
exit    (\)     Exit mysql. Same as quit
go      (\g)    Send command to mysql server
ego     (\G)    Send command to mysql server; Display result verically
print   (\p)    print current command
quit    (\q)    Quit mysql
rehash  (\#)    Rebuild completion hash
status  (\s)    Get status information from the server
use     (\u)    Use another database. Takes database name as argument

Connection id: 4  (Can be used with mysqladmin kill)

mysql>
```

Probieren wir nun einmal einige Befehle aus:

```
mysql> status
--------------
mysql  Ver 9.29 Distrib 3.22.16a-gamma, for pc-linux-gnu (i586)

Connection id:          4
Server version          3.21.33b
```

```
Protocol version        10
Connection              Localhost via UNIX socket
UNIX socket             /tmp/mysql.sock
--------------

mysql>
```

Wie man sehen kann, haben wir uns mit dem Aufruf von **mysql** schon an die Datenbank angebunden. Es wurde nach keinem Paßwort gefragt.

```
mysql> show databases; use mysql;
+----------+
| Database |
+----------+
| mysql    |
| test     |
+----------+
2 rows in set (0.01 sec)

ERROR 1044: Access denied for user: '@localhost' to database 'mysql'
mysql>
```

Wie man sehen kann, haben wir nun zwei Befehle an den MySQL Server gesendet. Der Zweite konnte nicht ausgeführt werden. **use** gibt an, auf welche Datenbank sich die DML Kommandos (Data Manipulation Language) sich beziehen sollen. Probieren wir also einen Zugriff auf die zweite Datenbank, test:

```
mysql> use test
Database changed
mysql>
```

Das ist fein ! Auf diese Datenbank haben wir also nun Zugriff. Wir lassen uns alle Tabellen anzeigen:

```
mysql> show tables;
Empty set (0.01 sec)

mysql>
```

Es sind keine Tabellen enthalten. Was stört, ist daß wir offensichtlich keine Administrator Rechte an dem Datenbankserver besitzen, da wir auf die Datenbank **mysql** offensichtlich keinen Zugriff haben. Schauen Sie hierzu im Kapitel Zugriffsberechtigungen (Privilege-System) oder im Kapitel Checkliste bei Zugriffsproblemen nach. Da dieses Phänomen hier sicher kein Einzelfall ist, schauen wir also im MySQL Handbuch nach, und finden die Lösung. Wir loggen uns mit **exit** aus und mit dem Shellbefehl: **mysql -u root mysql** wieder ein. Dann schauen wir uns den **Status** erneut an:

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8 to server version: 3.21.33b

Type 'help' for help.

mysql> status
--------------
mysql  Ver 9.29 Distrib 3.22.16a-gamma, for pc-linux-gnu (i586)

Connection id:          8
Current database:       mysql
Current user:           root
Server version          3.21.33b
Protocol version        10
Connection              Localhost via UNIX socket
UNIX socket             /tmp/mysql.sock
Uptime:                 2 hours 32 min 33 sec

Running threads: 1  Questions: 37  Opened_tables: 4  Flush tables: 1  Open
```

```
tables: 4
--------------

mysql>
```

Das ist schon besser. Wir erhalten nun zusätzliche Informationen über das System, die Datenbank mysql, u.s.w. Wir sind nun als root in der Datenbank mysql eingeloggt, und haben alle Rechte. Der Befehl SHOW sollte uns nun weiterhelfen, die Datenbank **mysql** zu ergründen. Probieren wir einmal einige SHOW-Befehle aus:

```
mysql> show databases;
+----------+
| Database |
+----------+
| mysql    |
| test     |
+----------+
2 rows in set (0.01 sec)

mysql> show tables;
+-----------------+
| Tables in mysql |
+-----------------+
| db              |
| func            |
| host            |
| user            |
+-----------------+
4 rows in set (0.01 sec)

mysql>
```

Aha ! Hier sieht die Welt nun so aus, wie in dem Kapitel GRANT oder im Kapitel Privilegien beschrieben. Diese beiden Kapitel sollte Sie sich nun zuerst einmal durchlesen, bevor es weitergehen kann.

```
mysql> show status;
+--------+-----------------+-----------+---------------+---------+-------------+
| Uptime | Running_threads | Questions | Opened_tables | Reloads | Open_tables |
+--------+-----------------+-----------+---------------+---------+-------------+
|  10000 |               1 |        47 |             4 |       1 |           4 |
+--------+-----------------+-----------+---------------+---------+-------------+
1 row in set (0.00 sec)

mysql>
```

Show Status zeigt mehr als nur **status**. Hier werden weitere Angaben zu den Tabellen der Datenbank **mysql** gemacht. Hier erfahren Sie auch, wie oft die Datenbank abgefragt wird, und wieviele simultane User auf diese Datenbank zugreifen. Was uns natürlich interessiert, sind die Zugriffsrechte, die in der Hosts-, User- und DB Tabelle abgelegt sind. Ja, MySQL hat das Rechtesystem einfach auch in Tabellen abgelegt. Sie sollten also die datenbank **mysql** nicht als irgendeine Datenbank betrachten, sondern als die Datenbank, die für die Rechte aller anderen Datenbanken zuständig ist, in unserem Fall also der leeren Datenbank **test**. Wie lassen wir uns nun die Inhalte des Rechtesystems ausgeben ? Die Antwort ist naheliegend - nämlich mit dem DML Befehl SELECT. Wir erinnern uns also an die Ausgabe von **show tables;**...und lassen uns die Inhalte aller Tabellen ausgeben: **select * from db,host,user;** wäre etwas unübersichtlich, also der Reihe nach:

```
mysql> select * from db;
+------+---------+------+-------------+-------------+-------------+------------>
| Host | Db      | User | Select_priv | Insert_priv | Update_priv | Delete_priv
+------+---------+------+-------------+-------------+-------------+------------>
| %    | test    |      | Y           | Y           | Y           | Y
| %    | test\_% |      | Y           | Y           | Y           | Y
+------+---------+------+-------------+-------------+-------------+------------>
2 rows in set (0.01 sec)
```

```
mysql> select * from host;
+-----------+----+-------------+-------------+-------------+-------------+-----+
| Host      | Db | Select_priv | Insert_priv | Update_priv | Delete_priv | Crea
+-----------+----+-------------+-------------+-------------+-------------+-----+
| localhost | %  | Y           | Y           | Y           | Y           | Y
| tunix     | %  | Y           | Y           | Y           | Y           | Y
+-----------+----+-------------+-------------+-------------+-------------+-----+
2 rows in set (0.01 sec)

mysql> select * from func;
Empty set (0.00 sec)

mysql>
```

Die Tabelle **user** können wir leider nicht darstellen, da sie einfach zu breit ist. Um sie als Ganzes sehen zu können, sollten Sie unter LINUX mit der KDE Oberfläche, 1024x768 Punkte Auflösung den Font Ihrer Shell verkleiner, indem Sie die **STRG** Taste festhalten und gleichzeitig mit der rechten Maustaste den Font **tiny** auswählen. Nun können Sie das Fenster vergrößern, und sich die Inhalte der Tabelle **user** nochmals ausgeben lassen:

```
mysql> select * from user;
+-----------+------+----------+-------------+-------------+-------------+-->
| Host      | User | Password | Select_priv | Insert_priv | Update_priv |
+-----------+------+----------+-------------+-------------+-------------+-->
| localhost | root |          | Y           | Y           | Y           | Y
| localhost |      |          | N           | N           | N           | N
| tunix     | root |          | Y           | Y           | Y           | Y
| tunix     |      |          | N           | N           | N           | N
+-----------+------+----------+-------------+-------------+-------------+--->
4 rows in set (0.00 sec)

mysql> select * from user;
```

Wir haben die Tabelle etwas abgeschnitten dargestellt. Es dürfte aber klar sein, was diese Tabelle enthält. Eine Liste der Einträge ist ja im Kapitel GRANT genau beschrieben. Die einzelnen Rechte werden in Spalten dargestellt. Können wir uns nun die Spalten einfach mal ausgeben lassen ?

Selbstverständlich ! Wir hätten gerne eine Beschreibung der Tabelle **host**. Das heißt auf englisch **beschreiben** ? Richtig, **describe** ! Also lautet der Befehl wie folgt:

```
mysql> describe host;
+-------------+---------------+------+-----+---------+-------+
| Field       | Type          | Null | Key | Default | Extra |
+-------------+---------------+------+-----+---------+-------+
| Host        | char(60)      |      | PRI |         |       |
| Db          | char(64)      |      | PRI |         |       |
| Select_priv | enum('N','Y') |      |     | N       |       |
| Insert_priv | enum('N','Y') |      |     | N       |       |
| Update_priv | enum('N','Y') |      |     | N       |       |
| Delete_priv | enum('N','Y') |      |     | N       |       |
| Create_priv | enum('N','Y') |      |     | N       |       |
| Drop_priv   | enum('N','Y') |      |     | N       |       |
+-------------+---------------+------+-----+---------+-------+
8 rows in set (0.00 sec)

mysql> describe db;
+-------------+---------------+------+-----+---------+-------+
| Field       | Type          | Null | Key | Default | Extra |
+-------------+---------------+------+-----+---------+-------+
| Host        | char(60)      |      | PRI |         |       |
| Db          | char(64)      |      | PRI |         |       |
| User        | char(16)      |      | PRI |         |       |
| Select_priv | enum('N','Y') |      |     | N       |       |
```

```
| Insert_priv | enum('N','Y') |      |     | N       |       |
| Update_priv | enum('N','Y') |      |     | N       |       |
| Delete_priv | enum('N','Y') |      |     | N       |       |
| Create_priv | enum('N','Y') |      |     | N       |       |
| Drop_priv   | enum('N','Y') |      |     | N       |       |
+-------------+---------------+------+-----+---------+-------+
9 rows in set (0.00 sec)

mysql> describe user;
+---------------+---------------+------+-----+---------+-------+
| Field         | Type          | Null | Key | Default | Extra |
+---------------+---------------+------+-----+---------+-------+
| Host          | char(60)      |      | PRI |         |       |
| User          | char(16)      |      | PRI |         |       |
| Password      | char(16)      |      |     |         |       |
| Select_priv   | enum('N','Y') |      |     | N       |       |
| Insert_priv   | enum('N','Y') |      |     | N       |       |
| Update_priv   | enum('N','Y') |      |     | N       |       |
| Delete_priv   | enum('N','Y') |      |     | N       |       |
| Create_priv   | enum('N','Y') |      |     | N       |       |
| Drop_priv     | enum('N','Y') |      |     | N       |       |
| Reload_priv   | enum('N','Y') |      |     | N       |       |
| Shutdown_priv | enum('N','Y') |      |     | N       |       |
| Process_priv  | enum('N','Y') |      |     | N       |       |
| File_priv     | enum('N','Y') |      |     | N       |       |
+---------------+---------------+------+-----+---------+-------+
13 rows in set (0.00 sec)

mysql> describe func;
+-------+------------+------+-----+---------+-------+
| Field | Type       | Null | Key | Default | Extra |
+-------+------------+------+-----+---------+-------+
| name  | char(64)   |      | PRI |         |       |
| ret   | tinyint(1) |      |     | 0       |       |
| dl    | char(128)  |      |     |         |       |
+-------+------------+------+-----+---------+-------+
3 rows in set (0.01 sec)

mysql>
```

Wir haben nun Informationen über die Tabellen erhalten, die etwas darüber hinausgehen, was wir im Moment benötigen. Wenn man sich die Tabellendefinitionen anschaut, so wird man feststellen, daß es Parallelen gibt. Es gibt offensichtlich Spalten, die die Tabellen miteinander verknüpfen. Diese haben in der Spalte **Key** den Ausdruck **PRI**, die Abkürzung für **Primary Key** stehen. Wie sich die Rechte für SQL Clients genau aus den drei Tabellen zusammensetzen, können Sie ja in den Kapiteln Privilegien und GRANT entnehmen. Von Interesse sollte auch der Default Wert in der 5 Spalte sein. Wann immer eine Tabelle oder eine Datenbank angelegt wird, dann werden die Rechte auf die Default Rechte gesetzt.

Nun wissen Sie also um das Geheimnis des schier endlosen Statements:

INSERT INTO user VALUES ('%', 'userxy', 'passwort', 'Y', 'Y', 'Y' .....);

Hierzu müssen Sie sich nur an die genaue Reihenfolge der Felder halten, die in **describe user;** ausgegeben wurden. Eine einfache Sache, insbesondere dann, wenn man bedenkt, daß man in der Kombination von MySQL und PHP3 oder PERL einfach mit Hilfe eines einfachen **INSERT** Statements Rechte an Inhalten der SQL Datenbank bestimmen kann. Einfacher geht´s nicht mehr ! Datenbanken renomierter Hersteller tun sich hierbei mitunter recht schwer.

So, nun kommen wir noch einmal zurück auf die Rechte an den Datenbanken **test** und **mysql**. Hierzu schauen wir uns noch einmal die Inhalte der Tabellen **db, user** und **host** (Alle Singular) an. Wir sehen, daß auf der Datenbank **test** keine Beschränkungen liegen, da hier fast alle Felder frei sind, oder ein **%**-Zeichen enthalten, was unter SQL gleichbedeutend mit einem Joker (*) ist. jeder darf auf die Datenbank **test** des MySQL Datenbankservers zugreifen. Kritisch ist allerdings der Zugriff auf die Datenbank **mysql**, da hiermit die komplette Rechtevergabe aller Datenbanken des MySQL Serverdämons verwaltet wird. Wir müssen also das Administratorpaßwort ändern....





6.5 Ändern des Administratorpaßwortes unter MySQL

Wir erinnern uns nun an die Default-Werte in den Tabellen db, host, und user. Wir müssen also damit rechnen, daß wenn wir eine Tabelle anlegen, jedermann/frau über einen Zugriff über das Netzwerk daraus Daten löschen oder diese verändern kann. Also Vorsicht.....

Da die Datenbank test schon einmal exisitiert, sollten wir nun vielleicht direkt einen User mit einem Paßwort einfügen.....Hierzu schauen wir uns das Beispiel aus dem Kapitel Beispiel zur Vergabe der Rechte an. An dieser Stelle sollte vielleicht noch kurzerwähnt werden, daß MySQL selbstverständlich auch den GRANT Befehl unterstützt, wie Sie es vielleicht von anderen Datenbanken gewöhnt sind. Der Befehl **INSERT INTO user VALUES .....** ist absolut gleichwertig. Alle Datenbankhersteller haben hier geringe Abweichungen bei der Bedeutung der Parameter des **GRANT** Statements, so auch MySQL. Das sollte man vielleicht bedenken, wenn man von oder auf andere Datenbanken wechselt.

Hier nun ein Statement:

```
mysql> insert into user values ('%', 'testuser', password('testpasswort'),
'Y', 'Y', 'Y', 'Y', 'Y', 'Y' ,'Y', 'Y', 'Y','Y');
Query OK, 1 row affected (0.00 sec)

mysql>
```

Ok, scheint funktioniert zu haben. Was haben wir den nun eigentlich verändert ? Wir haben einen **testuser** angelegt, der sich mit dem Paßwort **testpasswort** einloggen kann, und alle Rechte an der Datenbank **test** erhält. Wir überprüfen das einmal:

```
mysql> select * from user;
+-----------+----------+------------------+-------------+-------------+-->
| Host      | User     | Password         | Select_priv | Insert_priv |
+-----------+----------+------------------+-------------+-------------+-->
| localhost | root     |                  | Y           | Y           | Y
| localhost |          |                  | N           | N           | N
| tunix     | root     |                  | Y           | Y           | Y
| tunix     |          |                  | N           | N           | N
| %         | testuser | 34469d8918c0c98c | Y           | Y           | Y
+-----------+----------+------------------+-------------+-------------+-->
5 rows in set (0.00 sec)

mysql>
```

Wie man sehen kann, erfordert die Anmeldung als User **root** zwar ein Paßwort, das Feld ist aber leer. Das ist der Grund, warum wir uns bisher ohne Paßwort einloggen konnten. **MySQL** unterstützt beim Start des **mysqld** (Das Kommando **/sbin/init.d/mysql start** die Option **--password**. Siehe hierzu auch Kapitel Lösungen bei access denied Fehlern) Das Startskript von LINUX sorgt dafür, daß das Paßwort aus der Datei **/etc/mysql.passwd**, welches bei einer Standardinstallation von LINUX einfach leer ist, daß man ohne Paßwort den Datenbankserver MySQL administrieren kann. Wir loggen uns nun mit **exit** aus, und mit folgendem Statement wieder ein:

```
user01@tunix:~ > mysql -u testuser mysql
ERROR 1044: Access denied for user: '@localhost' to database 'mysql'
user01@tunix:~ > mysql -u testuser
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 29 to server version: 3.21.33b

Type 'help' for help.

mysql> quit
```

```
Bye
```

Das Einloggen als einfacher User **testuser** auf der Datenbank **mysql** funktioniert natürlich nicht. Schließlich haben wir den User testuser ja angelegt, damit er nur auf die Datenbank **test** zugreifen darf. Hier also der zweite, erfolgreiche Versuch:

```
user01@tunix:~ > mysql -h localhost -u testuser
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 32 to server version: 3.21.33b

Type 'help' for help.

mysql> status;
--------------
mysql  Ver 9.29 Distrib 3.22.16a-gamma, for pc-linux-gnu (i586)

Connection id:          32
Current database:
Current user:           testuser
Server version          3.21.33b
Protocol version        10
Connection              Localhost via UNIX socket
UNIX socket             /tmp/mysql.sock
Uptime:                 4 hours 47 min 9 sec

Running threads: 1  Questions: 151  Opened_tables: 4  Flush tables: 1  Open
tables: 4
--------------

mysql>
```

Wir haben es nun geschafft, uns als testuser anzumelden, allerdings haben wir einen Zugang, ohne daß wir das Paßwort angegeben haben:

```
user01@tunix:~ > mysql -h localhost -u testuser
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 33 to server version: 3.21.33b

Type 'help' for help.

mysql> use test
Database changed
mysql> show tables;
Empty set (0.02 sec)

mysql> use mysql;
ERROR 1044: Access denied for user: '@localhost' to database 'mysql'
mysql>
```

Nun müssen wir nur noch ein Paßwort für den User **root** vergeben, damit nicht jeder, der gerade einen **mysql** Client besitzt, beliebig sich in den Datenbankserver einloggen und herumadministrieren darf. Wir haben uns bisher ohne Paßwort an dem Datenbankserver anmelden können. Hierzu editieren wir als User **root** die Datei **/etc/mysql.passwd** und tragen dort unser neues Paßwort ein: **testadmin**. Danach starten wir die Datenbank neu:

```
bash-2.02# /sbin/init.d/mysql restart
Restarting service MySQL./usr/bin/mysqladmin: connect to server at
'localhost' failed
error: 'Access denied for user: 'root@localhost' (Using password: YES)' failed
bash-2.02#
```

Der Fehler ist so korrekt. Die von S.u.S.E. vorbereiteten Skripte, die sich nun mit einem Paßwort einloggen müssen, dürfen nun nicht mehr funktionieren, da wir ja das Paßwort in der **mysql** Tabelle noch nicht verändert haben. Damit keine

Mißverständnisse aufkommen. Das in der Datei **/etc/mysql.passwd** angegebene Paßwort hat nur Auswirkungen auf die Startup-Skripte, nicht aber auf das **root** Paßwort für den Datenbank-Server. Dieses werden wir nun ändern. Ähnlich dem **INSERT INTO...** Befehl ist auch der **REPLACE INTO** Befehl. **Ein kleiner Tip: Mit STRG-R kann man vorher eingegebene Befehle suchen und editieren:**

```
mysql> select * from user;
+-----------+----------+------------------+-------------+-------------+-->
| Host      | User     | Password         | Select_priv | Insert_priv |
+-----------+----------+------------------+-------------+-------------+-->
| localhost | root     |                  | Y           | Y           | Y
| localhost |          |                  | N           | N           | N
| tunix     | root     |                  | Y           | Y           | Y
| tunix     |          |                  | N           | N           | N
| %         | testuser | 34469d8918c0c98c | Y           | Y           | Y
+-----------+----------+------------------+-------------+-------------+-->
5 rows in set (0.00 sec)

mysql> replace into user values ('localhost', 'root', password('testadmin'),
'Y', 'Y', 'Y', 'Y', 'Y', 'N' ,'N', 'N', 'N','N');
Query OK, 2 rows affected (0.00 sec)

mysql>
```

Sie sollten sich darüber bewußt sein, daß hier viel zu viele Rechte vergeben wurden. Es ist Usern der Tabelle test sogar möglich, der MySQL Server anzuhalten, weil eventuell das **shutdown priv.** vergeben wurde. In der Praxis sollten insbesondere die letzten fünf 'Y' auf 'N' stehen. Damit können Daten verändert und Tabellen angelegt werden, jedoch ist es nicht möglich den Server herunterzufahren, oder andere Tabellen in der Datenbank zu löschen.

Sie können beim MySQL Client stets alle Befehle mit Hilfe der Cursortasten zurückholen, um längere Statements, wie das Beispiel **replace into...** nochmals zur Korektur von Zugriffsrechten aufzurufen. Alle Befehle werden in Ihrem Homeverzeichnis in der Datei **.mysql_history** gespeichert.

Damit die Änderungen in der Datenbank wirksam werden, müssen wir aus dem MySQL Client aussteigen, die Datenbank neu laden, und uns wieder einloggen:

```
mysql> quit
Bye
user01@tunix:~ > mysqladmin -u root reload
user01@tunix:~ > mysql -u root mysql
ERROR 1045: Access denied for user: 'root@localhost' (Using password: NO)
user01@tunix:~ >
```

Offensichtlich hat es funktioniert. Das Einloggen ohne Paßwort funktioniert nicht mehr ! Probieren wir es also mit Paßwort:

```
user01@tunix:~ > mysql -u root -ptestadmin mysql
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 42 to server version: 3.21.33b

Type 'help' for help.

mysql> status;
--------------
mysql  Ver 9.29 Distrib 3.22.16a-gamma, for pc-linux-gnu (i586)

Connection id:          42
Current database:       mysql
```

```
Current user:           root
Server version          3.21.33b
Protocol version        10
Connection              Localhost via UNIX socket
UNIX socket             /tmp/mysql.sock
Uptime:                 5 hours 43 min 44 sec

Running threads: 1  Questions: 223  Opened_tables: 4  Flush tables: 1  Open
tables: 4
--------------

mysql>
```

Es hat funktioniert ! Von nun an ist unser MySQL Datenbankserver gegen unbefugten Zugriff gesichert. Niemand kann nun auf das Rechtesystem, also die Datenbank **mysql** ohne das gültige Paßwort zugreifen ! Allerdings ist unsere Datenbank **test** noch für Zugriffe offen. Nun müßte das Skript **/sbin/init.d/mysql restart** auch ohne Probleme funktionieren, da das interne Paßwort nun ja mit dem Paßwort in der Datei **/etc/mysql.passwd** übereinstimmt. Unser Server ist nun korrekt aufgesetzt. Nun möchten wir jedoch diesen nicht von der Konsole aus administriren, sondern eventuell auch einmal von unserer Windows Arbeitsstation aus.... Aktivieren Sie nun das Powermanagement für CPU und Festplatte, entfernen Sie Monitor und Tastatur und stellen Sie Ihren LINUX MySQL Server in die Ecke, sie benötigen keinen Zugang mehr zum System. Alles weitere läuft nun über Fernadministration.






6.6 Remote Administration des MySQL Servers

Zur Fernadministration des MySQL Servers von irgendeiner Arbeitsstation benötigen wir die Administrationswerkzeuge, die wir bereits kennengelernt haben, jedoch als Win32 Version. Laden Sie sich diese entweder aus unserem Supportverzeichnis: http://www.rent-a-database.de/support/mysql/clients, oder installieren Sie sich die Win32 Version von MySQL auf Ihrer Arbeitsstation möglichst im Verzeichnis **c:\mysql**. Die Clients befinden sich dann im Verzeichnis **c:\mysql\bin**. Sie haben sich doch sicher die IP-Nummer und die Portnummer Ihres MySQL Servers aufgeschrieben. Nun benötigen wir diese Angaben. Setzen Sie die IP-Nummer Ihres MySQL Servers ein und beachten Sie genau auch die Leerzeichen:

```
c:\mysql\bin>mysql -h 10.0.0.5 -u testuser -ptestpasswort test
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 62 to server version: 3.21.33b

Type 'help' for help.

mysql>
```

Wir können nun auch von unserer Windows Arbeitsstation aus die Datenbank **test** administrieren. Die Datenbank **mysql** läßt sich nicht von der Arbeitsstation aus administrieren, da von dem Client immer der User **root@10.0.0.1** an den Server übermittelt wird. Das dient dazu, daß man in den Logdateien von MySQL immer sehen kann, von welchem Host aus zugegriffen wird. Damit wir uns als Administrator einloggen können, muß ein User **root@ip-nummer** mit der IP-Nummer der Arbeitsstation in die **host** Tabelle der Datenbank **mysql** eingetragen werden. Vorerst kümmern wir uns nicht mehr darum und widmen uns den verschiedenen Clients.....




6.7 Vorstellung der Clients für MySQL

Den Client **mysql**, der bei allen MySQL Distributionen beiliegt, haben wir nun in der Version für UNIX (LINUX) und Windows 98/NT kennengelernt. Dieser wird im Prinzip nur dann gebraucht, wenn man z.B. Skript im Batch-Modus, siehe Kapitel Batch-Modus ausführen möchte.

In unserer Support-Datenbank auf http://www.rent-a-database.de/support/mysql/win32/clients/ finden Sie weitere Clients. Installieren Sie bitte das Programm **MySqlAdministrator** auf Ihrer Arbeitsstation:

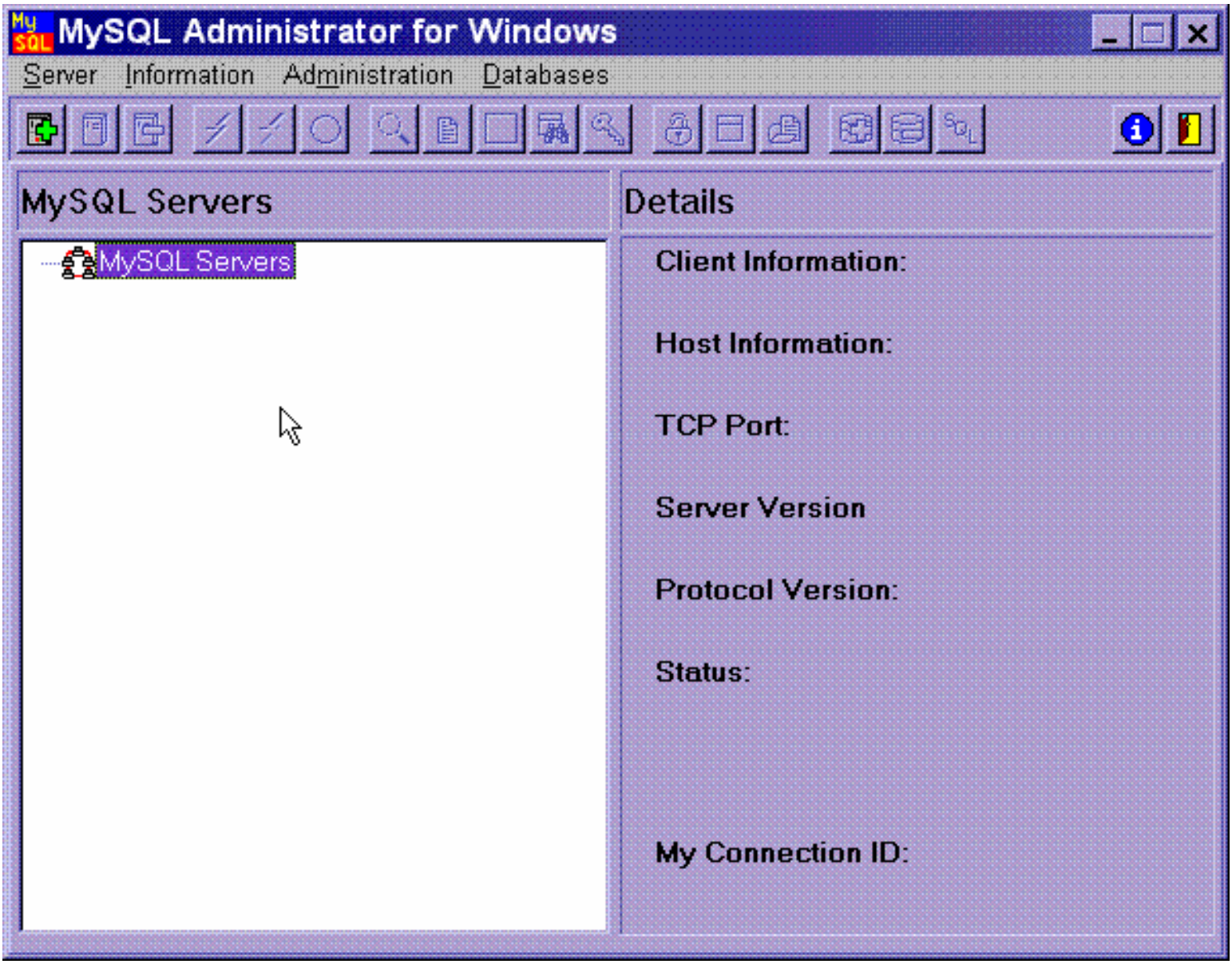


Abbildung:MySQL Client

Klicken Sie links oben auf den Button mit dem grünen Kreuz (Add Server). Es öffnet sich ein Fenster. Hier tragen Sie (von oben nach unten) einen Namen Ihrer Wahl, die IP-Nummer Ihres Datenbankservers und den **Login User Namen**, in unserem Fall also **testuser** ein. Für den Fall, daß Sie MySQL 3.23 installiert haben, müssen die im letzen Feld die Portnummer von 3306 auf 3333 abändern. Klicken Sie auf OK. Sie sehen nun im linken Feld unter MySQL Servers einen Server mit dem Namen Ihrer Wahl

angezeigt:

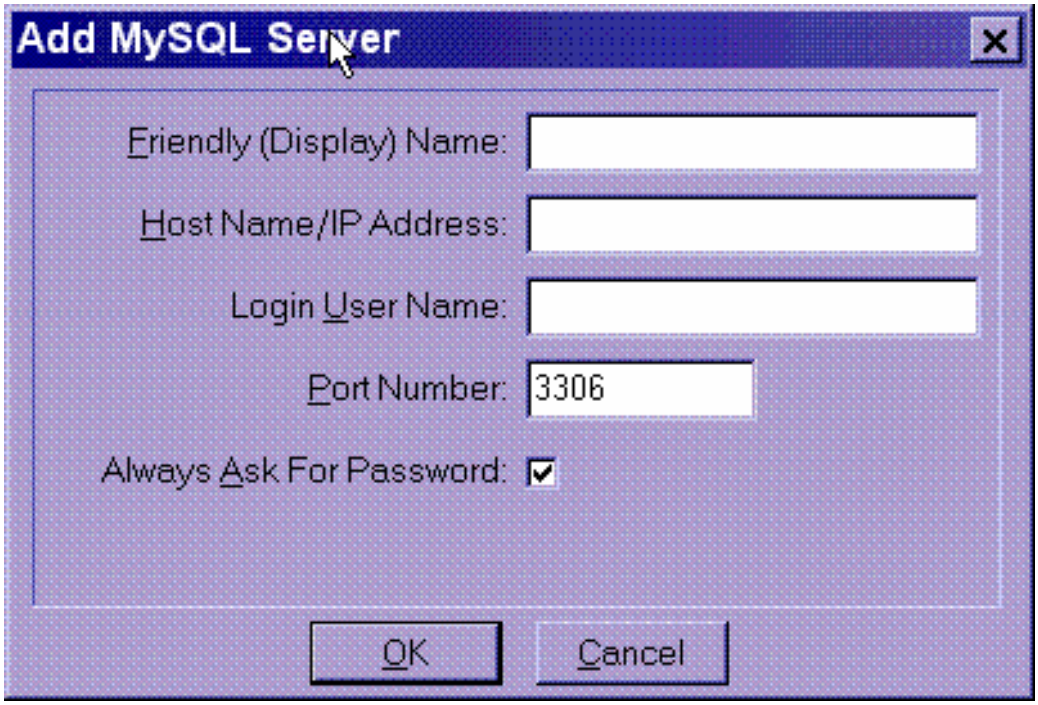


Abbildung:ACCESS2

Klicken Sie nun auf den Blitz oben in der Menüleiste, um eine Verbindung herzustellen:

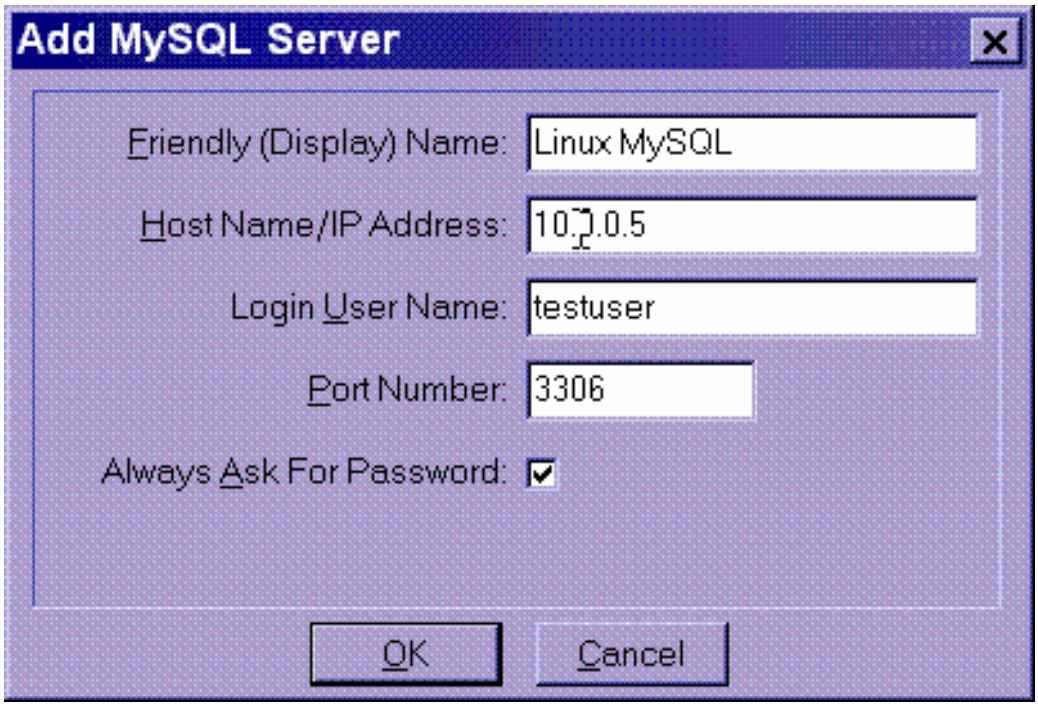


Abbildung:ACCESS3

Es öffnet sich ein Fenster, in welchem nach dem Paßwort gefragt wird. Geben Sie hier **testpasswort** ein.



Abbildung:004

Klicken Sie nun auf das kleine Kreuzchen neben dem Servernamen. Es öffnet sich nun ein komfortables Menü, in dem alle unsere uns wohlbekannten Datenbanken mit allen Tabellen angezeigt werden:

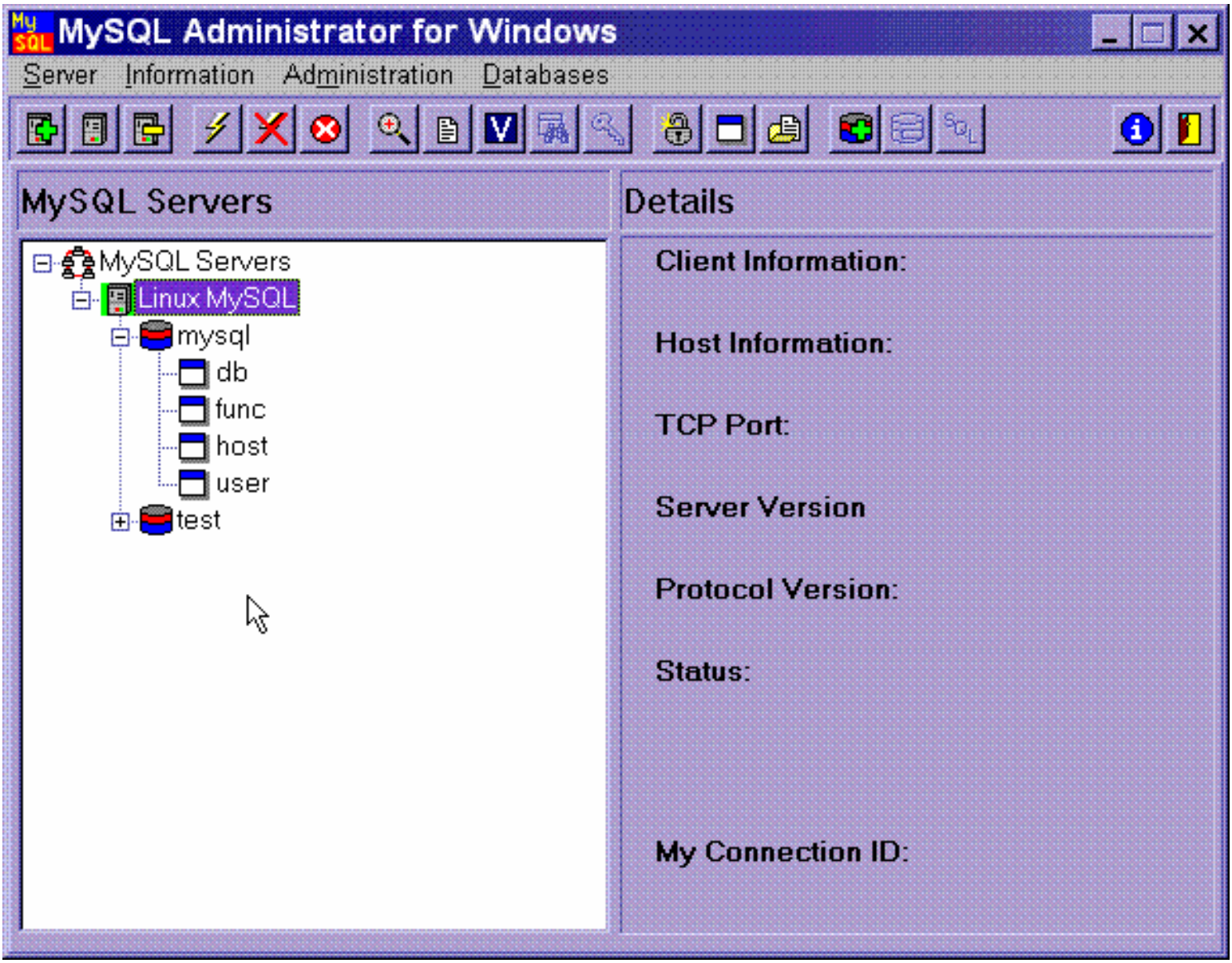


Abbildung:004

Sie haben nun die Verbindung zu dem MySQL Server hergestellt und können nun ebenso, wie mit dem originalen MySQL Client Befehle an den Datenbankserver absetzen.

Testen Sie alle bisherigen SQL Statements aus. Hierzu klicken Sie auf den **SQL** Button rechts oben, oder Sie öffnen mit der rechten Maustaste ein SQL Fenster.

Probieren Sie einmal einige Statements aus:

select * from host

Beachten Sie, daß nun die Kommandos nicht mehr mit einem Semikolon oder **\g** abgeschlossen werden müssen. Administrator-Rechte haben Sie nach dem Einloggen als **testuser** nicht, was Sie daran sehen können, daß im Menü Administration das Untermenü **Reload GRANT Tables** und im Menü Databases das Menü Create/Drop Database inaktiv ist. Sie dürfen nun aber in der Datenbank **test** Tabellen anlegen. Hierzu klicken Sie auf den Button **SQL** und geben folgendes ein:

```
create table testtabelle (spalte1 integer, spalte2 char(20))
```

Beachten Sie bitte auch das fehlende Semikolon am Ende des CREATE Statements !

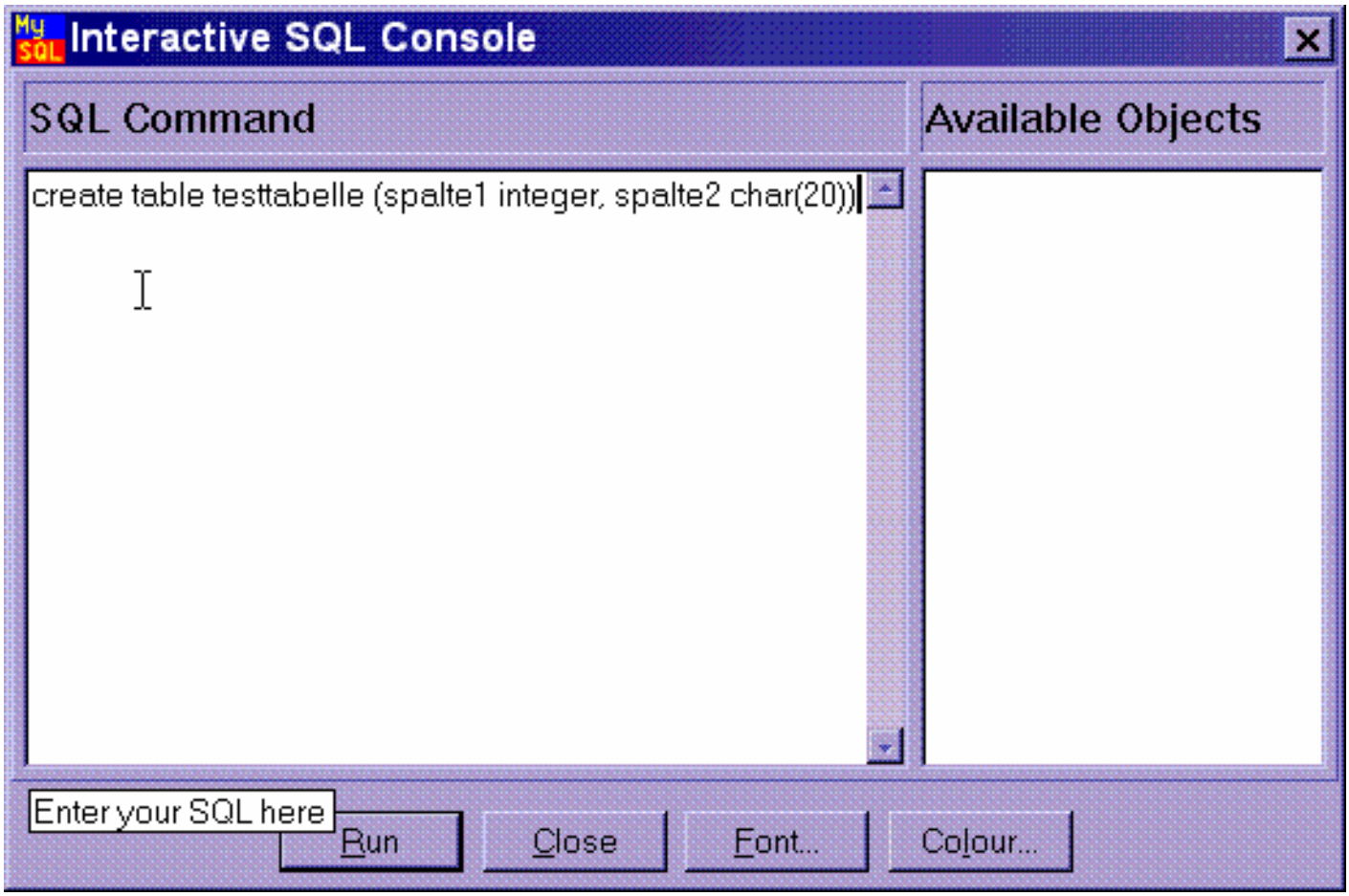


Abbildung:005

Obwohl das Kommando akzeptiert wurde, werden die Änderungen noch nicht auf dem Client angezeigt. Hierzu müssen Sie wieder auf **Linux MySQL** klicken, damit die Menüs aktiviert werden, und dann auf den blauweißen Button **Refresh Database Tables** klicken. Danach müssen Sie die Verbindung zur Datenbank beenden (Button Disconnect) und sich wieder anbinden (Button Connect). Danach erst werden alle Änderungen auch dargestellt. Der Grund liegt in dem fortgeschrittenen THREAD Konzept. Für jede Verbindung und auch Abfrage wird ein eigener Thread erzeugt. Damit der Client von den Änderungen erfährt, muß man dafür sorgen, daß ein neuer Thread mit frischen Informationen den Client versorgt:

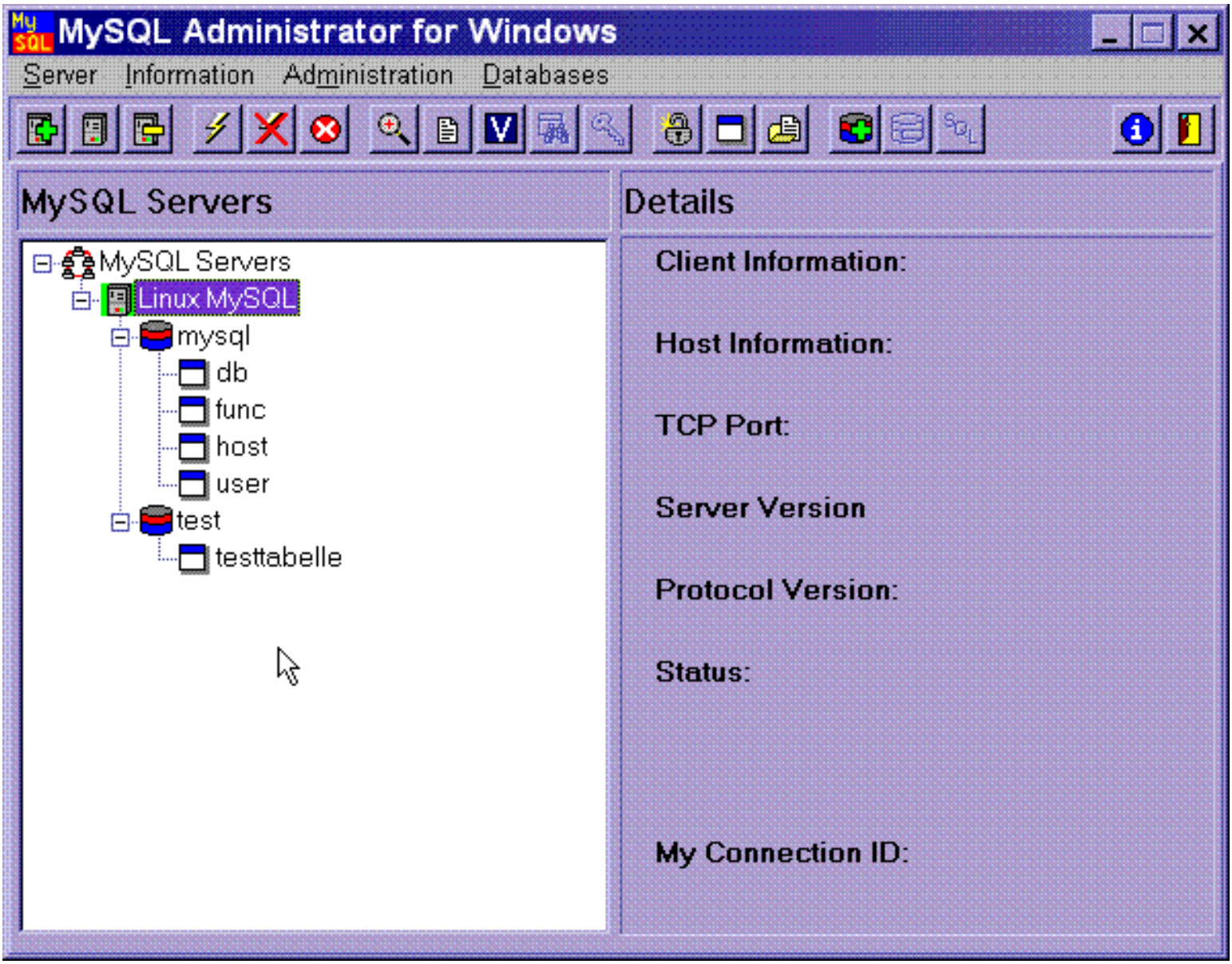


Abbildung:007

Zugegeben, es fehlen viele der sinnvollen Optionen, die man bei jeder Spalte stets mit definieren sollte, jedoch geht es uns zuerst darum, eine Datenbank zu erstellen, zu füllen und abzufragen. Schauen Sie hierzu bitte in das Kapitel CREATE. Nun weiter mit dem nächsten Kapitel....

6.8 Datensatz in die Tabelle einfügen

Während das SQL Statement **create** noch zu der DDL (Data Definition Language) gehört, so wenden wir uns nun der DML, der Data Manipulation Language zu. Hierzu fügen wir Daten in die gerade angelegte Tabelle ein und lesen diese wieder aus. Hierzu bedienen wir uns wiederum unseres bewährten Toolkits. Damit das SQL Fenster rechts oben aktiviert wird, müssen Sie zuerst die Datenbank wählen, auf die sich das Statement beziehen soll. Klicken Sie also zuerst auf die Datenbank **test**. Sie können so mit diesem Toolkit tausende Datenbanken und hunderte von MySQL Servern verwalten. Geben Sie nun also in das SQL Fenster folgendes ein (ohne Semikolon in dem Fenster, mit Semikolon, falls Sie den normalen MySQL Client verwenden:

```
mysql> insert into testtabelle (spalte1, spalte2) values(5,"testwert");

Query OK, 1 row affected (0.00 sec)

mysql>
```

Klicken Sie einmal auf Datenbank **test** und dort auf die Tabelle **testtabelle**. Es öffnet sich ein neues Fenster, in der Sie die Struktur der Tabelle **testtabelle** sich genau anschauen können:

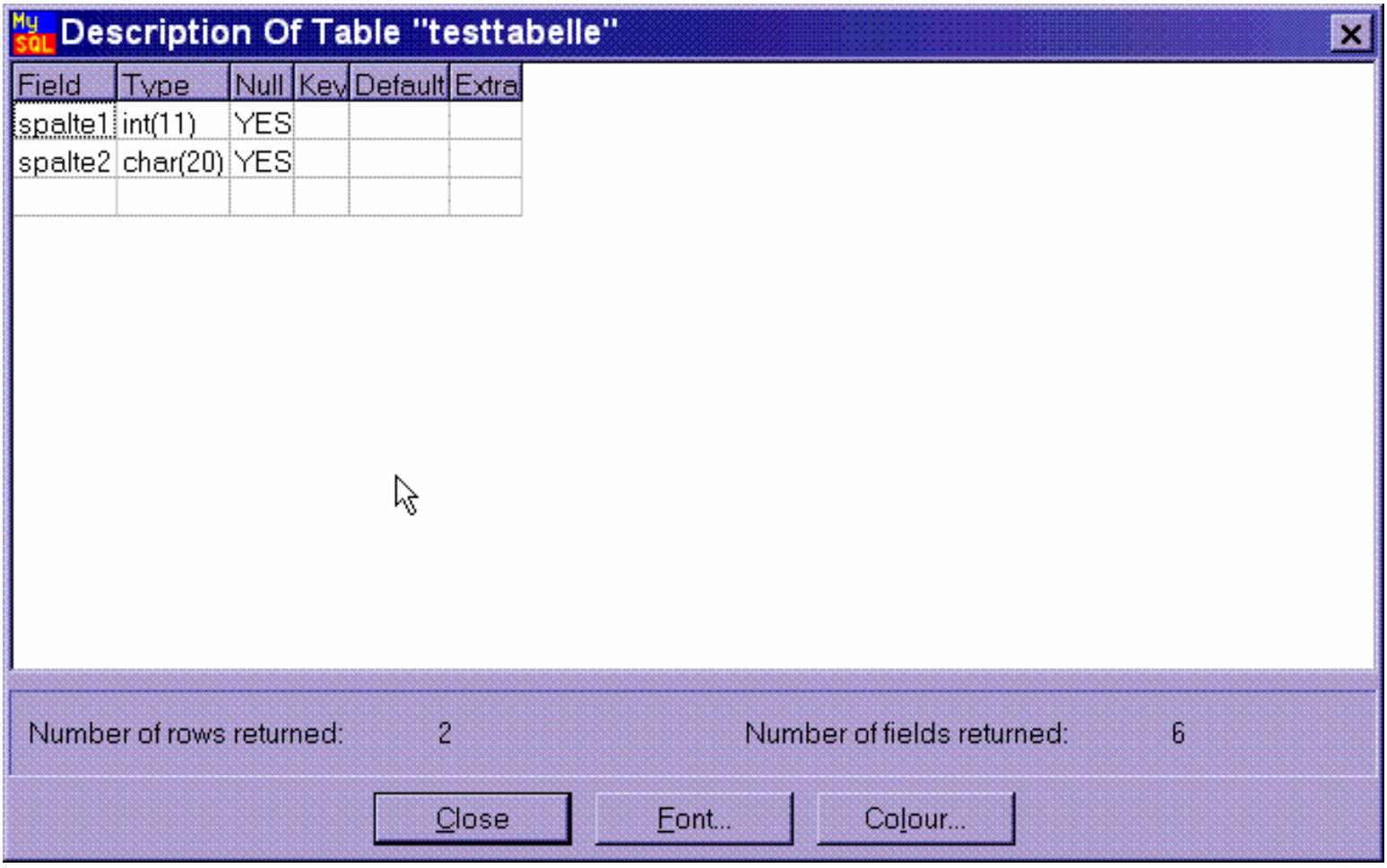


Abbildung:008

Fügen Sie nun Daten in die Tabelle über den MySQL Client ein. Hierzu öffnen Sie wieder das SQL Fenster und geben das SQL Statement wie zuvor ein:

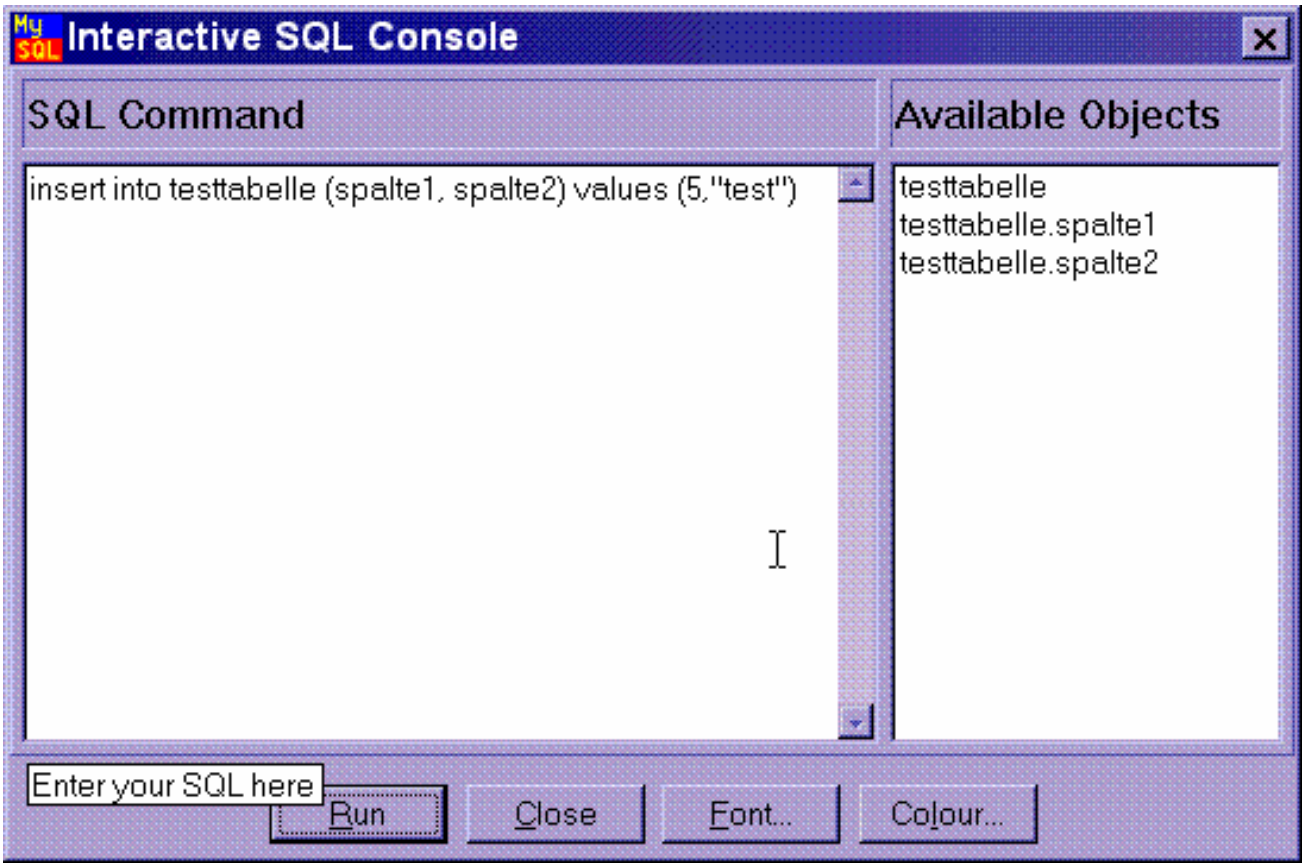


Abbildung:009

Klicken Sie auf **"RUN"**. Sie sehen nun im Fenster das Ergebnis dessen, was der MySQL Serverdämon zurückmeldet. Hier wird nicht mehr die Zeit ausgegeben, wie beim originalen MySQL Client, jedoch sehen Sie hier, wieviele Zeilen in der Tabelle von Ihrem Statement betroffen waren:

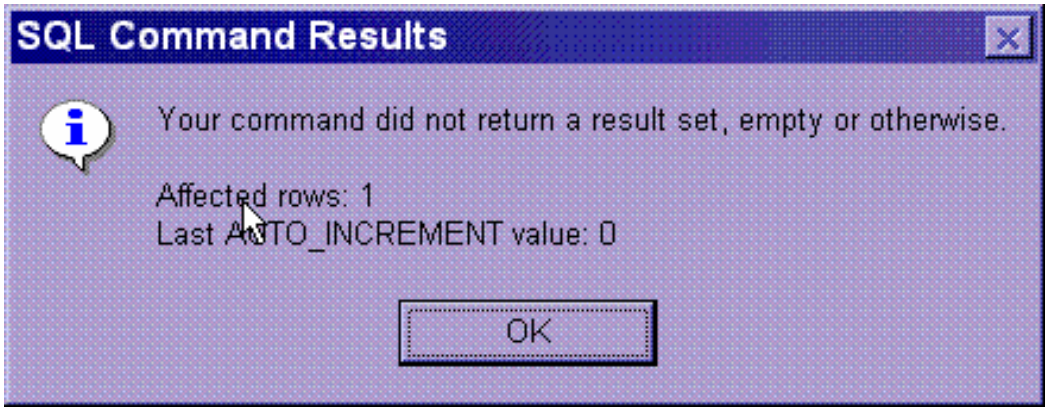

SQL Command Results
Your command did not return a result set, empty or otherwise.
Affected rows: 1
Last AUTO_INCREMENT value: 0
OK

Abbildung:010

Wir lesen nun die gerade eingefügten Daten aus der Testtabelle wieder aus:

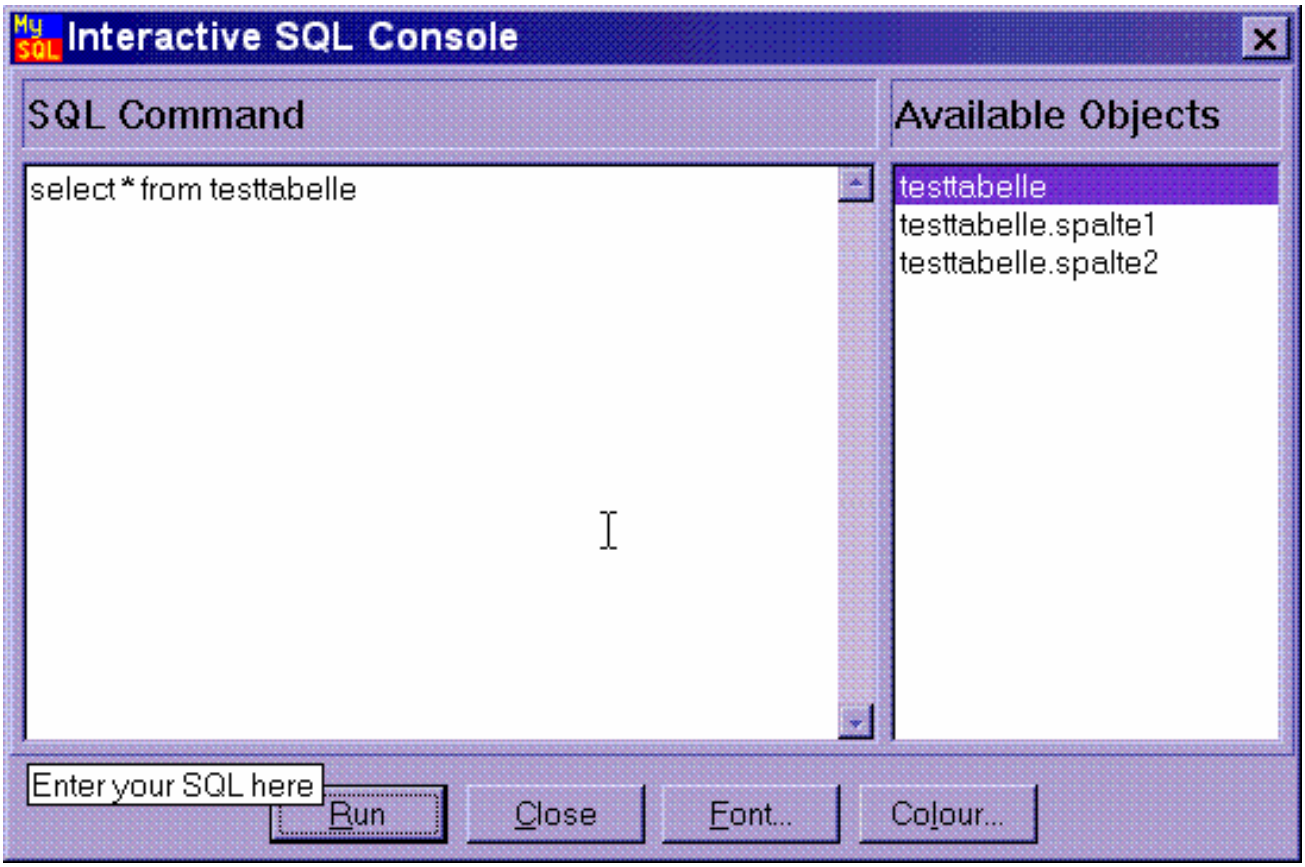


Abbildung:012

Hier nun die Ausgabe:

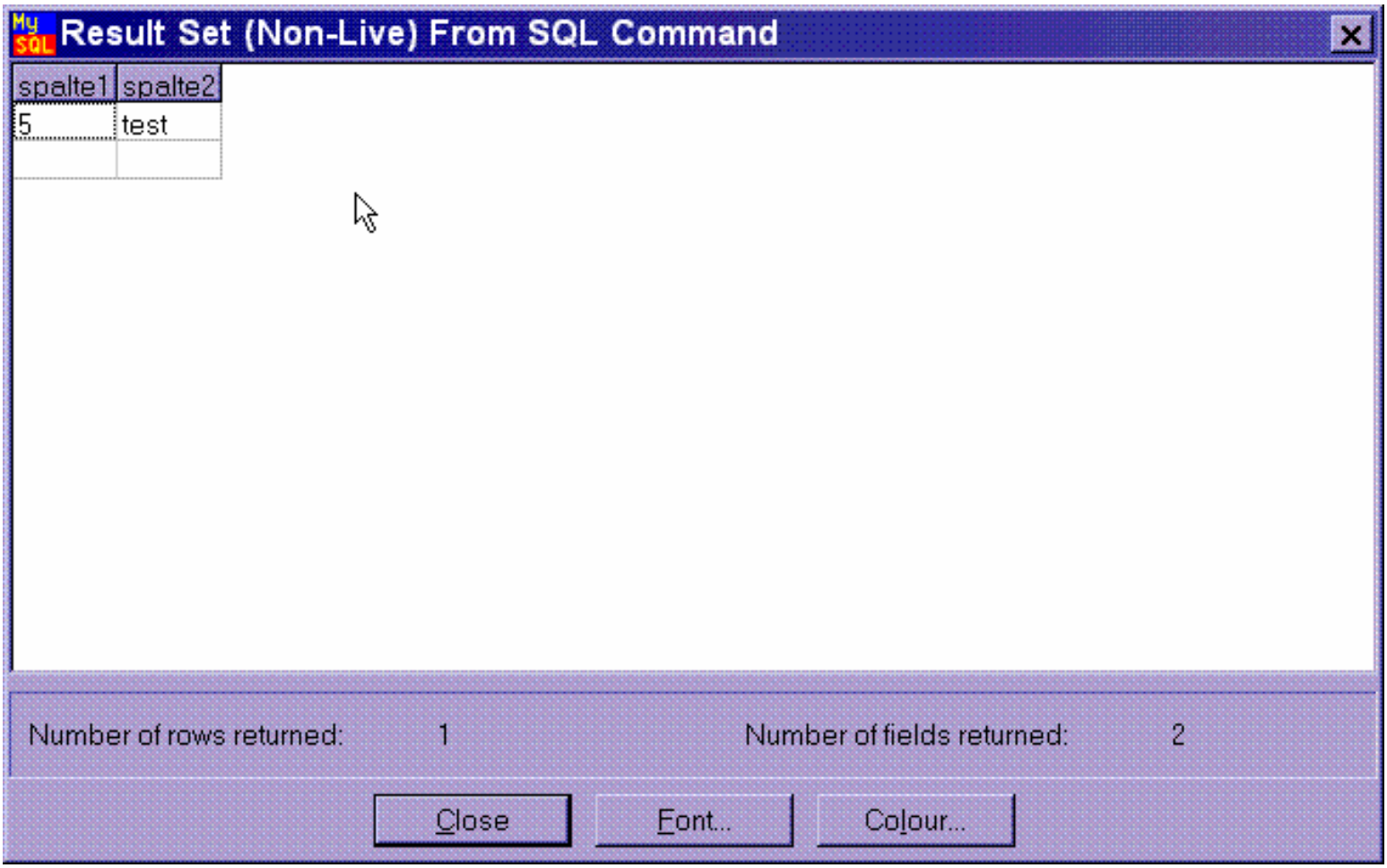


Abbildung:012

Bezüglich Groß/Kleinschreibung muß man festhalten, daß alle in MySQL reservierten Worten es keinen Unterschied macht, ob diese groß oder klein geschrieben sind. Bei Namen von Tabelle, Spalten u.s.w. spielt Groß/Kleinschreibung eine wichtige Rolle. Man sollte auch bedenken, daß es eventuell später bei einer Konvertierung in ein anderes Datenbankformat das konvertierende Programm vielleicht Groß/Kleinschreibung nicht unterscheiden kann. LINUX, SUN SOLARIS und NT sind UNICODE fähig, jedoch gibt es bei NT viele historisch bedingte Probleme. MySQL ist ebenfalls UNICODE fähig, jedoch muß man MySQL hierzu neu kompilieren, damit auch andere Schriften, wie KANJI, u.s.w. gespeichert werden können.

Bei den Inhalten unterscheidet MySQL in den meisten Fällen nicht zwischen Groß-Kleinschreibung. Dieses Verhalten ist sowohl abhängig von den Spracheinstellungen von MySQL, als auch von dem Datentyp. Nur bei den Datentypen, wie z.B. **BLOB** muß zwangsläufig exakt verglichen werden, da dieser Datentyp zur Speicherung von Binärdaten gedacht ist. Bei allen anderen Datentypen wird bei der Abfrage nicht nach Groß-/Kleinschreibung unterschieden. Das ist in fast allen Fällen auch sinnvoll so, insbesondere wenn man bedenkt, daß man oft auch nach Substrings suchen kann. So ist es z.B. bei der Suche nach einem Herrn **Maier** auch sinnvoll, eventuell nach **Meyer** oder **Kleinmayer** suchen zu können. In dem letzten Fall wird das **m** klein geschrieben.





6.9 Exportieren von Datensätzen aus MySQL

Im Moment sind ja kaum Daten in der Tabelle enthalten. Der Grund, warum das Kapitel **Exportieren ....** vorgezogen wird, ist derjenige, daß wir erst einmal schauen möchten, in welchem Datenformat die Daten abgelegt werden. Es ist logisch, daß MySQL diese Daten dann auch wieder importieren kann. Schauen wir uns etwas im Kapitel Sprachreferenz MySQL um. Logischerweise muß man zuerst Daten auswählen, die dann exportiert werden können, daher muß der Befehl also im Kapitel SELECT zu finden sein....genau **SELECT FROM table into outfile "name";** Wir probieren es zunächst aus gutem Grund mit dem LINUX Client auf dem MySQL Server:

```
user01@tunix:~/SGML > mysql -h 10.0.0.5 -u testuser -ptestpasswort test
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10 to server version: 3.21.33b

Type 'help' for help.

mysql> select * from testtabelle into outfile "datenexport.txt";
Query OK, 2 rows affected (0.03 sec)

mysql> quit
```

Wir merken uns, daß alle Dateinamen, Strings u.s.w. stets in Anführungszeichen stehen müssen.

Schön, wo sind die Daten nun hin ? Wie sehen Sie aus ? Wir können feststellen, daß die Daten nicht, wie zunächst vermutet, im Verzeichnis des MySQL Client unter LINUX sind. MySQL hat beim Kompilieren immer ein Verzeichnis angegeben bekommen, in welchem standardmäßig immer alle Datenbanken versammelt auf der Festplatte gespeichert sind. Dies ist das Verzeichnis **/var/mysql**. Hier finden sich alle unsere Datenbanken und Tabellen wieder:

```
user01@tunix:/var/mysql > ls -la
total 9
drwxr-xr-x   4 root     root         1024 Aug 27 07:10 .
drwxr-xr-x  24 root     root         1024 Mar 12 17:49 ..
drwxr-xr-x   2 root     root         1024 Aug 19 10:26 mysql
drwxr-xr-x   2 root     root         1024 Aug 27 09:18 test
-rw-r--r--   1 root     root          794 Aug 27 07:10 tunix.err
-rw-r--r--   1 root     root         2418 Aug 27 07:10 tunix.log
-rw-r--r--   1 root     root            3 Aug 27 07:10 tunix.pid
user01@tunix:/var/mysql >
```

Schauen wir uns nun im Verzeichnis **test** um:

```
user01@tunix:/var/mysql > cd test
user01@tunix:/var/mysql/test > ls -la
total 14
drwxr-xr-x   2 root     root         1024 Aug 27 09:18 .
```

```
drwxr-xr-x   4 root     root         1024 Aug 27 07:10 ..
-rw-rw-rw-   1 root     root           18 Aug 27 09:18 datenexport.txt
-rw-rw----   1 root     root           52 Aug 27 07:41 testtabelle.ISD
-rw-rw----   1 root     root         1024 Aug 27 07:41 testtabelle.ISM
-rw-rw----   1 root     root         8590 Aug 27 07:12 testtabelle.frm
user01@tunix:/var/mysql/test >
```

Aha, logischerweise speichert MySQL die Exportdateien ebenfalls in dem Verzeichnis ab, in dem sich die MySQL Datenbank befindet.....Schauen wir uns die Datenbank einmal an.....

```
user01@tunix:/var/mysql/test > more datenexport.txt
5       test
5       testwert
user01@tunix:/var/mysql/test >
```

Fein ! Wir fügen also einfach mit dem Editor **joe** ein paar Daten hinzu. An dieser Stelle sollten wir vielleicht die wichtigsten Kommandos von **joe** erklären:

- joe dateiname.txt startet den Editor zum Bearbeiten der Datei
- STRG-K x dient dem Speichern und beenden
- STRG-K r dem Einlesen einer Datei
- STRG-K h blendet die Hilfe ein und aus
- Die Tabulatortaste sollten Sie verwenden....

Mehr gibt´s nicht zu wissen. Fügen Sie nun einige Spalten hinzu:

```
5       test
5       testwert
34567   kannix und istnix weissnix habenix
```

Beim Speichern erscheint vielleicht eine Fehlermeldung (could not make backup file, save anyway ?). Das ist so korrekt, wenn Sie nur als User eingeloggt sind. Die Rechte an der Datei selber haben Sie, jedoch Sie haben kein Recht, Dateien mit Ihrem Useraccount anzulegen. Es funktioniert aber trotzdem. Diese veränderten Daten sollten wir probeweise einmal einlesen. Hierzu verwenden wir das Statement aus unserer MySQL Sprachreferenz **load data infile...**, siehe hierzu auch Kapitel LOAD DATA INFILE.

```
user01@tunix:~/SGML > mysql -h 10.0.0.5 -u testuser -ptestpasswort test
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 11 to server version: 3.21.33b

Type 'help' for help.

mysql> load data infile "datenexport.txt" into table testtabelle;
Query OK, 3 rows affected (0.13 sec)
Records: 3  Deleted: 0  Skipped: 0  Warnings: 1

mysql>
```

Wir überprüfen einmal....

```
mysql> select * from testtabelle;
+---------+----------------------+
| spalte1 | spalte2              |
```

```
+---------+----------------------+
|       5 | test                 |
|       5 | testwert             |
|       5 | test                 |
|       5 | testwert             |
|   34567 | kannix und istnix we |
+---------+----------------------+
5 rows in set (0.00 sec)

mysql>
```

Tja, irgendwie hat es ja funktioniert, leider haben wir nun ein Problem, was wahrscheinlich viele teilen. Daher haben wir es hier auch provoziert.....

Es wurden Daten beim Import abgeschnitten. Dies ist ein recht häufiges Problem. Man stellt fast immer erst später fest, daß es den einen oder anderen Datensatz gibt, der doch länger ist, als man vorher bei der Definition der Tabelle gedacht hat. Zur Erinnerung: Wir hatten die Länge des Feldes **spalte2** mit **char(20)** angegeben.....zuwenig !

Also sollten wir unsere MySQL Sprachreferenz einmal wieder bemühen.....**ALTER** ist im Abschnitt ALTER zu finden. **ALTER table MODIFY** sollte passen ! Wir werden nach der Veränderung der Spalte die Daten nochmals importieren und den Tabelleninhalt schließlich ausgeben:

```
mysql> alter TABLE testtabelle modify spalte2 char(30);
ERROR 1064: parse error near 'modify spalte2 char(30)' at line 1
mysql>
mysql> alter TABLE testtabelle change  spalte2 spalte2 char(40);
Query OK, 3 rows affected (0.01 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> load data infile "datenexport.txt" into table testtabelle;
Query OK, 3 rows affected (0.01 sec)
Records: 3  Deleted: 0  Skipped: 0  Warnings: 0

mysql> select * from testtabelle;
+---------+-------------------------------------+
| spalte1 | spalte2                             |
+---------+-------------------------------------+
|       5 | test                                |
|       5 | testwert                            |
|   34567 | kannix und istnix we                |
|       5 | test                                |
|       5 | testwert                            |
|   34567 | kannix und istnix weissnix habenix  |
+---------+-------------------------------------+
6 rows in set (0.00 sec)

mysql>
```

Aus irgendeinem Grund hat MySQL die Syntax mit MODIFY nicht akzeptiert. Schauen wir noch einmal genau in die Sprachreferenz von MySQL. Offensichtlich ist diese Syntax erst seit der Version 3.22.16a möglich, da hier ein Tribut an ORACLE gemacht wurde. MODIFY ist kein ANSI SQL 92 Ausdruck, jedoch häufig verwandt, da ORACLE quasi als Standard angesehen werden kann. MySQL unterstützt viele ORACLE Klauseln, LOAD DATA INFILE ... ist z.B. eine solche, die nur ORACLE und MySQL beherrschen. Damit man Datensätze beim Einlesen überschreiben kann, hat das Statement **LOAD DATA LOCAL INFILE INTO tabelle ....** noch eine Option **REPLACE**, die die aktuelleren Datensätze aus der Importdatei in der Tabelle

ersetzt. Das funktioniert nur in Verbindung mit UNIQUE Keys.. .Offensichtlich hat es mit dem Ausdruck CHANGE funktioniert, allerdings sollten Sie beachten, daß der Spaltenname doppelt angegeben werden muß:

mysql> alter TABLE testtabelle change spalte2 spalte2 char(40);

Danach haben wir die Daten noch einmal eingelesen. Bitte beachten Sie, daß SQL hier Daten hinzugefügt hat, ohne zu bemerken, daß diese bereits in der Datenbank enthalten sind. Das ist ein häufiger Fehler, aber kein Problem. Dafür gibt es die DELETE Syntax, mit der wir die Tabelle vor dem Einlesen der Daten von der Festplatte löschen können:

```
delete from testtabelle;  load data infile "datenexport.txt" into table
testtabelle; select * from testtabelle;
Query OK, 0 rows affected (0.00 sec)

    -> testtabelle; select * from testtabelle;
Query OK, 3 rows affected (0.00 sec)
Records: 3  Deleted: 0  Skipped: 0  Warnings: 0

+---------+-------------------------------------+
| spalte1 | spalte2                             |
+---------+-------------------------------------+
|       5 | test                                |
|       5 | testwert                            |
|   34567 | kannix und istnix weissnix habenix  |
+---------+-------------------------------------+
3 rows in set (0.00 sec)

mysql>
```

Wie man nun sehen kann, darf man viele Statements in eine Zeile schreiben, solange man jedes Statement mit einem Semikolon beendet und somit von anderen Statements abtrennt. Das Ergebnis ist wie gewünscht.






6.10 Import von Datensätzen in MySQL

Im Kapitel Export von Datensätzen wurden bereits zum Test die Datensätze eingelesen. Bitte schauen Sie bezüglich der Syntax dort oder im Unterkapitel LOAD DATA INFILE nach.

Das Beispiel in dem vorangegangenen Kapitel erlaubte es uns, aus dem Standardverzeichnis von MySQL Daten einzulesen. Netterweise kann man aber auch folgendes Konstrukt angeben. Hierzu kopieren Sie bitte die Datei **datenexport.txt** aus dem Standard Verzeichnis in Ihr Homeverzeichnis (bitte anpassen !):

```
user01@tunix:~ > cp /var/mysql/test/datenexport.txt .
```

Sie löschen nun die Inhalte der Tabelle **testtabelle** und importieren die Daten:

```
mysql> delete from testtabelle;
Query OK, 0 rows affected (0.00 sec)

mysql> load data infile "/home/user01/datenexport.txt" into table testtabelle;
Query OK, 3 rows affected (0.01 sec)
Records: 3  Deleted: 0  Skipped: 0  Warnings: 0

mysql> select * from testtabelle;
+---------+-------------------------------------+
| spalte1 | spalte2                             |
+---------+-------------------------------------+
|       5 | test                                |
|       5 | testwert                            |
|   34567 | kannix und istnix weissnix habenix  |
+---------+-------------------------------------+
3 rows in set (0.00 sec)

mysql>
```

Ahnen Sie etwas ? Offensichtlich kann man auch absolute Pfade angeben, um Daten aus Nachbarverzeichnissen oder aus geschützten Bereichen in die Datenbank zuladen oder zu speichern !

Probieren wir es aus:

```
user01@tunix:~ > mysql -h 10.0.0.5 -u testuser -ptestpasswort test

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 17 to server version: 3.21.33b

Type 'help' for help.

mysql> create table passwd (text char(100));
Query OK, 0 rows affected (0.00 sec)

mysql> load data infile "/etc/passwd" into table passwd;
Query OK, 31 rows affected (0.01 sec)
Records: 31  Deleted: 0  Skipped: 0  Warnings: 0

mysql> select * from passwd;
+-----------------------------------------------------------------------+
| text                                                                  |
+-----------------------------------------------------------------------+
```

```
| root:x:0:0:root:/root:/bin/bash                                          |
| bin:x:1:1:bin:/bin:/bin/bash                                             |
| daemon:x:2:2:daemon:/sbin:/bin/bash                                      |
| lp:x:4:7:lp daemon:/var/spool/lpd:/bin/bash                              |
| news:x:9:13:News system:/etc/news:/bin/bash                              |
| uucp:x:10:14::/var/lib/uucp/taylor_config:/bin/bash                      |
.....
| nobody:x:65534:65534:nobody:/tmp:/bin/bash                               |
| user01:x:500:100::/platte2/home/user01:/bin/bash                         |
| user02:x:501:100::/home/user02:/bin/bash                                 |
| user03:x:502:100::/home/user03:/bin/bash                                 |
+--------------------------------------------------------------------------+
31 rows in set (0.00 sec)

mysql>
```

Man kann offensichtlich in einem S.u.S.E. LINUX System mit einem MySQL Server beliebig Daten aus anderen Verzeichnissen in die MySQL Datenbank einlesen und sich quer über das Internet irgendwohin übertragen lassen. Ein Useraccount mit beschränkten Rechten an irgendeiner Tabelle reicht da völlig aus. Sie sehen, daß da auch eine Firewall nicht mehr helfen kann.

Das Statement:

```
insert .....
select * from passwd into outfile "/etc/passwd";
Query OK, 32 rows affected (0.03 sec)
```

spare ich mir nun...

Damit ein solcher Einbruch also nicht möglich ist, sollten Sie MySQL nur mit User-Rechten starten, und den MySQL Serverdämon in eine CHROOT() Umgebung verbannen. Wie Sie das tun, finden Sie hier: http://www.little-idiot.de/firewall.

Ich kann Ihnen aber versichern, daß man LINUX und auch MySQL sicher so installieren kann, daß diese einfachen Tricks nicht mehr funktionieren. Falls Sie also eine gesicherte Datenbank im Internet betreiben wollen, wenden Sie sich an http://www.rent-a-database.de.








6.11 Einlesen von Datensätzen über das Netzwerk

Nun haben wir bisher nur die Möglichkeit kennengelernt, eine Datei einzuladen, die bereits auf dem Server abgelegt ist. Eine Datei, z.B. aus EXCEL, ACCESS oder DBASE als ASCII exportiert, könnte man zuerst auf den Server kopieren, und dann mit dem Statement **LOAD DATA INFILE ... INTO TABLE ...** importieren. Es gibt jedoch viel elegantere Methoden ....:

Update mit LOAD DATA LOCAL...

In MySQL 3.23 gibt es jedoch die Möglichkeit, mit dem original MySQL Client von Ihrer Arbeitsstation aus eine Datei über das Netzwerk, das kann auch ISDN sein, in die Datenbank einzulesen. Benutzen Sie hierzu den Client aus von unserem Server http://www.rent-a-database.de/mysql/client/:

```
mysql> load data local infile "c:\datenexport.txt" into table testtabelle;
Query OK, 3 rows affected (0.00 sec)
Records: 3  Deleted: 0  Skipped: 0  Warnings: 0

mysql> select * from testtabelle;
+---------+-------------------------------------+
| spalte1 | spalte2                             |
+---------+-------------------------------------+
|       5 | test                                |
|       5 | testwert                            |
|   34567 | kannix und istnix weissnix habenix  |
|       5 | test                                |
|       5 | testwert                            |
|   34567 | kannix und istnix weissnix habenix  |
+---------+-------------------------------------+
6 rows in set (0.00 sec)

mysql>
```

Fein, hat also funktioniert. Damit Sie die Daten auch ersetzen, während Sie die Daten updaten, sollten in der Tabelle eine Spalte im einem UNIQUE KEY enthalten sein. Sie können dann mit folgendem Statement Ihr Update durchführen, während Ihre Datenbank weiterhin abgefragt wird. Hier nun obiges Statement mit der Option **replace**. Versetzen Sie hierzu Ihre Datenbank zuvor wieder in den Ausgangszustand:

```
mysql> load data local infile "c:\datenexport.txt" replace into table testtabelle;
Query OK, 3 rows affected (0.00 sec)
Records: 3  Deleted: 0  Skipped: 0  Warnings: 0

mysql> select * from testtabelle;
+---------+-------------------------------------+
| spalte1 | spalte2                             |
+---------+-------------------------------------+
|       5 | test                                |
|       5 | testwert                            |
|   34567 | kannix und istnix weissnix habenix  |
+---------+-------------------------------------+
3 rows in set (0.00 sec)

mysql>
```

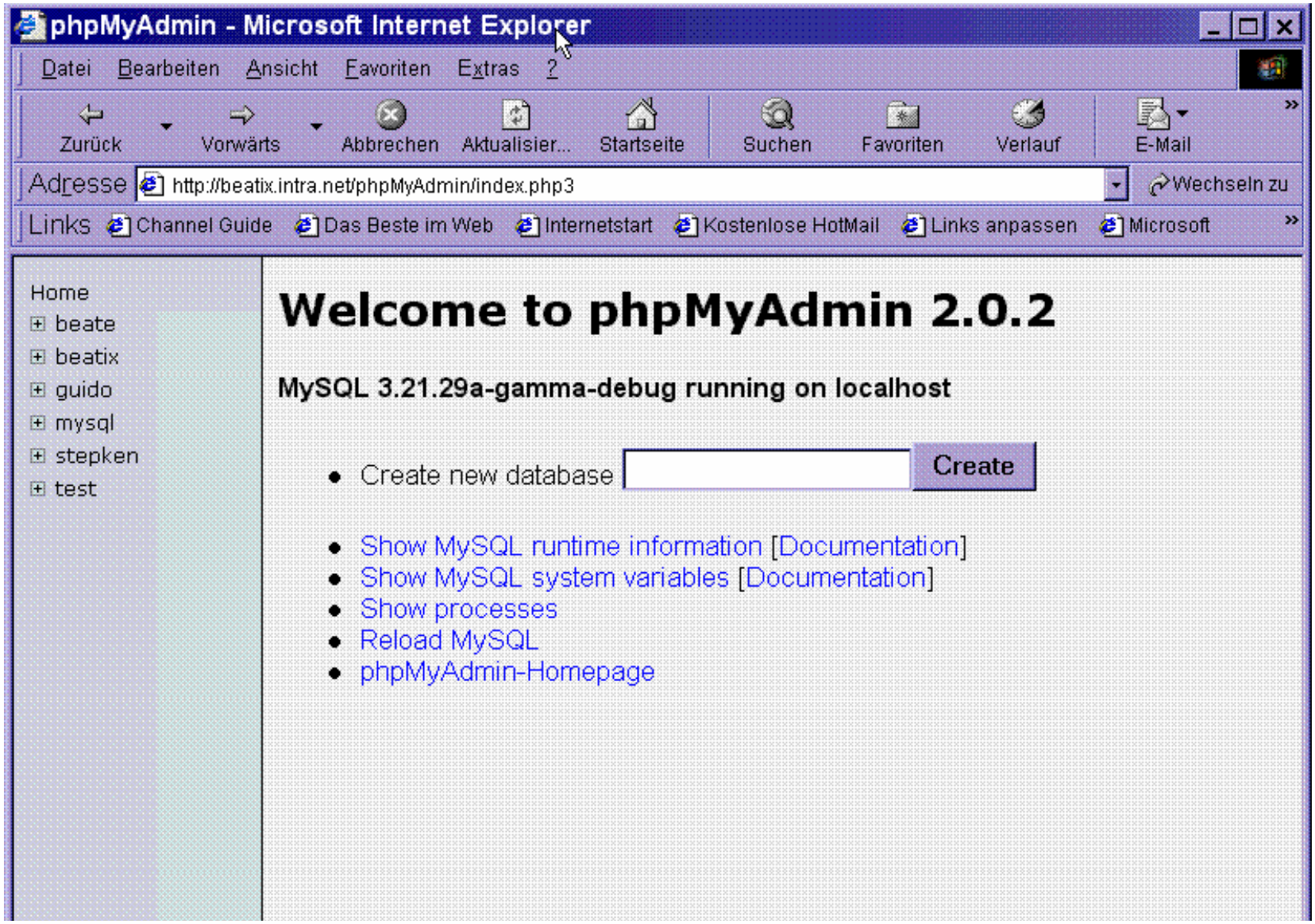
Es hat offensichtlich funktioniert !




6.12 Administration über PHP3

Zur Installation von MySQL, Apache und PHP3 auf Ihrer Arbeitsstation lesen Sie bitte Kapitel Windows, Apache, MySQL und PHP3.

Wir gehen nun im folgenden davon aus, daß Sie MySQL in c:\mysql und das Administrationswerkzeug phpMyAdmin 2.0.x im Unterverzeichnis **phpMyAdmin** des Apache **Documentroot** installiert haben. Starten Sie nun MySQL und Apache auf Ihrer lokalen Arbeitsstation mit http://localhost/phpMyAdmin/index.php3.



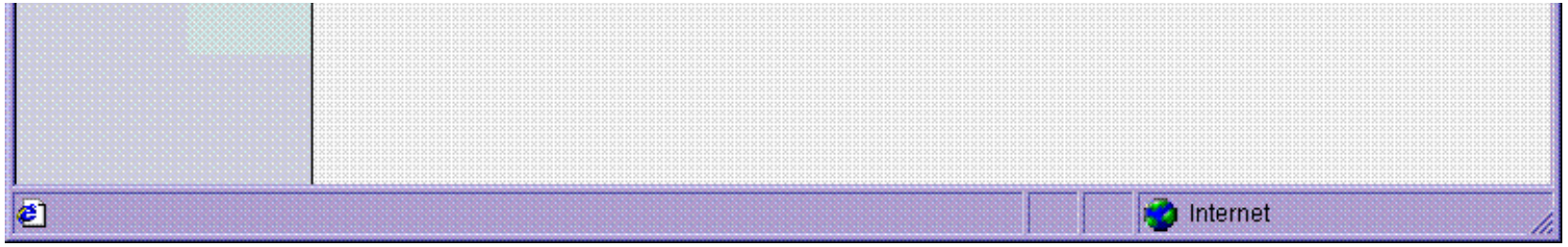


Abbildung:phpMyAdmin 2.0.x

Das phpMyAdmin Toolkit ist nun aktiv. Es verbindet sich über das LOOPBACK Interface von Windows 98/NT mit dem lokalen MySQL Serverdämon. Wir schauen uns nun einmal die Ausgabe von **netstat -n** und **ipconfig**in der DOS Shell von Windows 98 an:

```
C:\>netstat -n

Aktive Verbindungen

  Proto  Lokale Adresse          Remote-Adresse          Status
  TCP    10.0.0.1:1037           10.0.0.5:23             ESTABLISHED
  TCP    127.0.0.1:1050          127.0.0.1:3306          TIME_WAIT
  TCP    127.0.0.1:3306          127.0.0.1:1049          TIME_WAIT

C:\>ipconfig

Windows 98 IP-Konfiguration

0 Ethernet Adapter :

        IP-Adresse. . . . . . . . . : 10.0.0.1
        Subnet Mask . . . . . . . . : 255.255.255.0
        Standard-Gateway . . . . . . :

1 Ethernet Adapter :

        IP-Adresse. . . . . . . . . : 0.0.0.0
        Subnet Mask . . . . . . . . : 0.0.0.0
        Standard-Gateway . . . . . . :

C:\>route print

Aktive Routen:

  Netzwerkadresse      Subnet Mask  Gateway-Adresse    Schnittstelle  Anzahl
        10.0.0.0    255.255.255.0        10.0.0.1        10.0.0.1       1
        10.0.0.1  255.255.255.255       127.0.0.1       127.0.0.1       1
   10.255.255.255  255.255.255.255        10.0.0.1        10.0.0.1       1
```

```
          127.0.0.0        255.0.0.0        127.0.0.1        127.0.0.1       1
          224.0.0.0        224.0.0.0         10.0.0.1         10.0.0.1       1
    255.255.255.255  255.255.255.255         10.0.0.1         10.0.0.1       1

  C:\>
```

Nein, das ist nicht UNIX, auch wenn es so aussieht. Microsoft hat versucht, mit Windows 95/98 und NT das Betriebssystem UNIX zu Clonen, ist allerdings noch weit davon entfernt.... Probieren Sie unter LINUX einmal **ifconfig, netstat -a**, und **route -n** aus und Sie werden ähnliche Dinge sehen. Wir wissen nun, daß auch das **phpMyAdmin 2.0.x** Skript über das LOOPBACK Interface an den MySQL Server auf der Arbeitsstation anbindet. Wir können als diesem Mitteilen, daß es eine andere IP-Nummer ansprechen soll. Damit können wir von der lokalen Arbeitsstation jeden MySQL Server im Intranet oder auch Internet administrieren. PHP3 etwickelt sich quasi zu einer SWISS ARMY KNIFE Solution, wie die Amerikaner sagen...

Sofern Sie alle Installationsanweisungen korrekt befolgt haben , können Sie nun mit dem Wordpad die Datei **c:\apache\htdocs\phpMyadmin\config.inc.php3** öffnen:

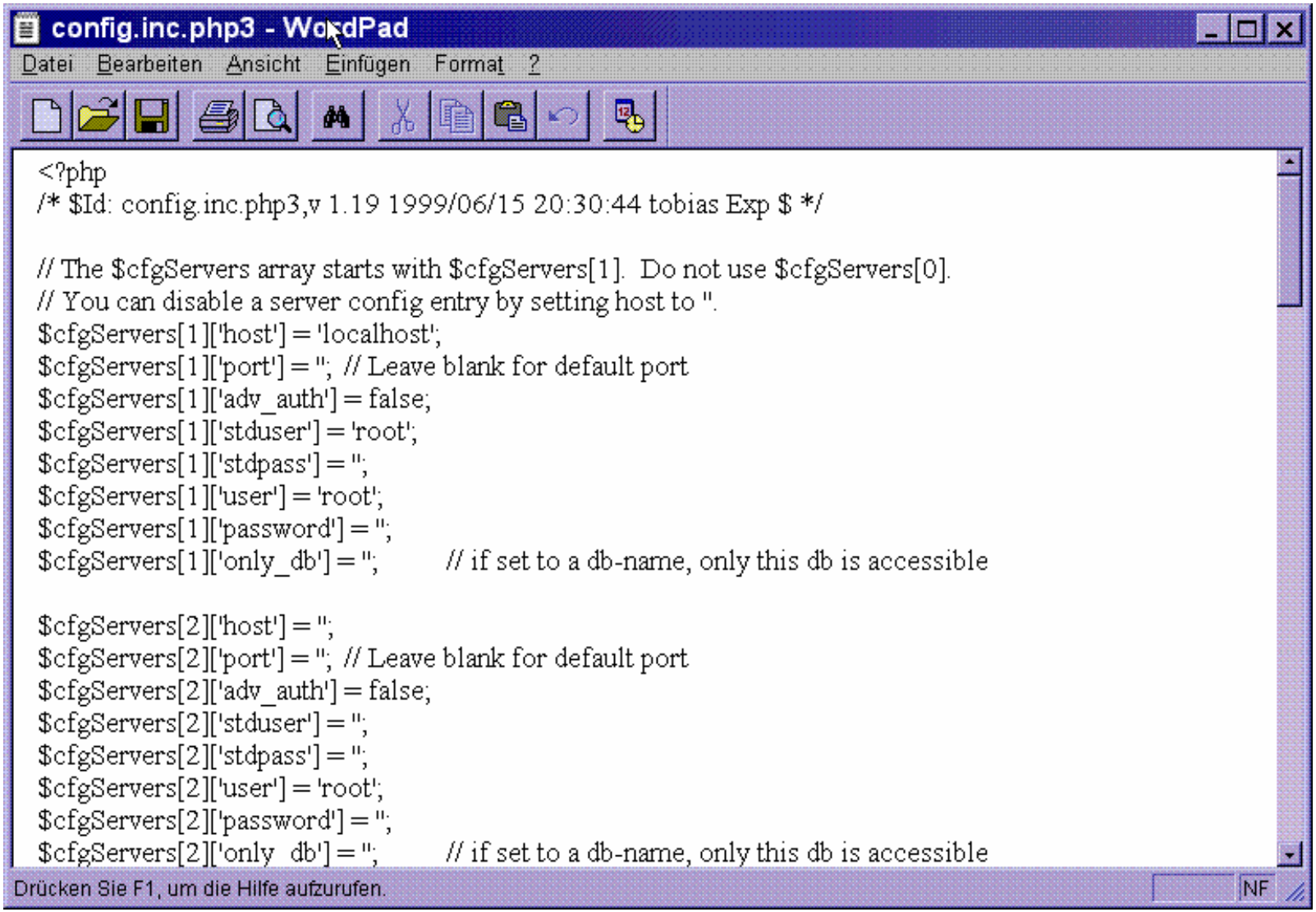

config.inc.php3 - WordPad
Datei Bearbeiten Ansicht Einfügen Format ?
<?php
/* $Id: config.inc.php3,v 1.19 1999/06/15 20:30:44 tobias Exp $ */
// The $cfgServers array starts with $cfgServers[1]. Do not use $cfgServers[0].
// You can disable a server config entry by setting host to ".
$cfgServers[1]['host'] = 'localhost';
$cfgServers[1]['port'] = ''; // Leave blank for default port
$cfgServers[1]['adv_auth'] = false;
$cfgServers[1]['stduser'] = 'root';
$cfgServers[1]['stdpass'] = '';
$cfgServers[1]['user'] = 'root';
$cfgServers[1]['password'] = '';
$cfgServers[1]['only_db'] = ''; // if set to a db-name, only this db is accessible
$cfgServers[2]['host'] = '';
$cfgServers[2]['port'] = ''; // Leave blank for default port
$cfgServers[2]['adv_auth'] = false;
$cfgServers[2]['stduser'] = '';
$cfgServers[2]['stdpass'] = '';
$cfgServers[2]['user'] = 'root';
$cfgServers[2]['password'] = '';
$cfgServers[2]['only_db'] = ''; // if set to a db-name, only this db is accessible
Drücken Sie F1, um die Hilfe aufzurufen.
NF

Abbildung:mysql016

Tragen Sie hier die IP-Nummer und die Portnummer Ihres MySQL LINUX Servers anstelle des lokalen MySQL Servers oder diejenige unserers Schulungs-Servers im Internet ein. Desweiteren müssen Sie sich nun entscheiden, ob Sie sich als **root** mit dem Paßwort **testadmin**, oder als User **testuser** mit dem Paßwort **testpasswort** anmelden wollen. Falls Sie Fragen zu den Bedeutungen der Felder haben, lesen Sie im Zweifelsfalle hierzu die Datei **C:\apache\htdocs\phpMyAdmin\Documentation.html** durch. Hier sehen Sie ein Beispiel, welches ohne Probleme funktionieren sollte:

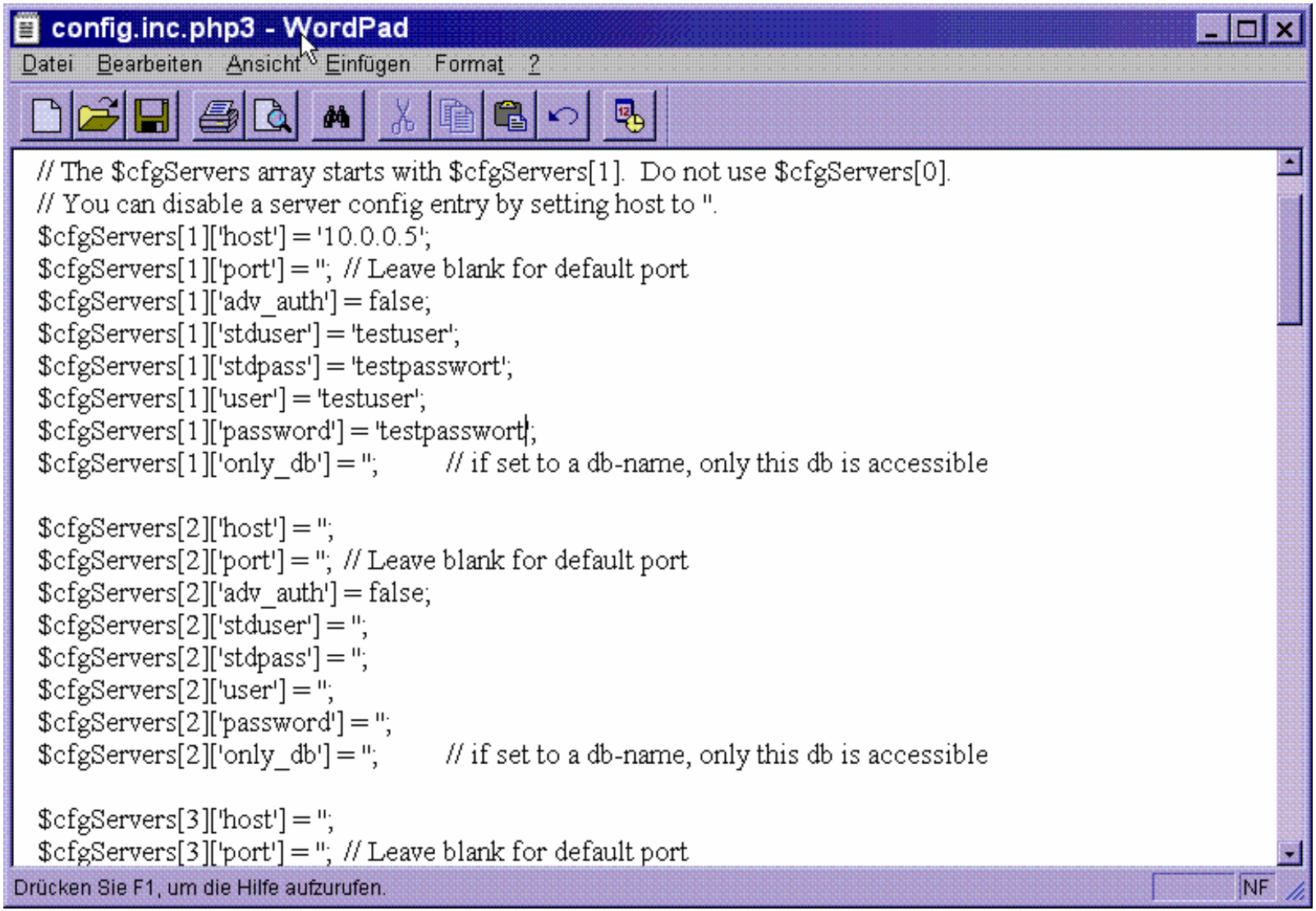

config.inc.php3 - WordPad
Datei Bearbeiten Ansicht Einfügen Format ?
// The $cfgServers array starts with $cfgServers[1]. Do not use $cfgServers[0].
// You can disable a server config entry by setting host to ''.
$cfgServers[1]['host'] = '10.0.0.5';
$cfgServers[1]['port'] = ''; // Leave blank for default port
$cfgServers[1]['adv_auth'] = false;
$cfgServers[1]['stduser'] = 'testuser';
$cfgServers[1]['stdpass'] = 'testpasswort';
$cfgServers[1]['user'] = 'testuser';
$cfgServers[1]['password'] = 'testpasswort';
$cfgServers[1]['only_db'] = ''; // if set to a db-name, only this db is accessible
$cfgServers[2]['host'] = '';
$cfgServers[2]['port'] = ''; // Leave blank for default port
$cfgServers[2]['adv_auth'] = false;
$cfgServers[2]['stduser'] = '';
$cfgServers[2]['stdpass'] = '';
$cfgServers[2]['user'] = '';
$cfgServers[2]['password'] = '';
$cfgServers[2]['only_db'] = ''; // if set to a db-name, only this db is accessible
$cfgServers[3]['host'] = '';
$cfgServers[3]['port'] = ''; // Leave blank for default port
Drücken Sie F1, um die Hilfe aufzurufen.
NF

Abbildung:018

Nun kehren Sie entweder mit der **Zurück Taste** Ihres Browsers zu dem Eingangsbildschirm des phpMyAdmin Skriptes zurück, und klicken Sie auf **Aktualisieren**, oder geben Sie **http://localhost/phpMyAdmin/index.php3** nocheinmal ein. Es könnte eventuell bei **localhost** ein wenig dauern, ansonsten verwenden sie die IP-Nummer Ihrer Arbeitsstation, die Sie sich mit **ipconfig** ja haben gerade ausgeben lassen....

Es öffnet sich nun der Eingangsbildschirm von **phpMyAdmin** mit einer kleinen Änderung - Sie können nun zwischen verschiedenen Servern wählen, die Sie administrieren möchten, falls Sie einen zweiten Server hinzugefügt haben. Wir haben ja nur die Adressen des lokalen Servers gegen diejenigen unseres LINUX Servers ausgetauscht. Sie sollten nun nach einem **Aktualisieren** wieder unsere Datenbanken **mysql** und **test** sehen:

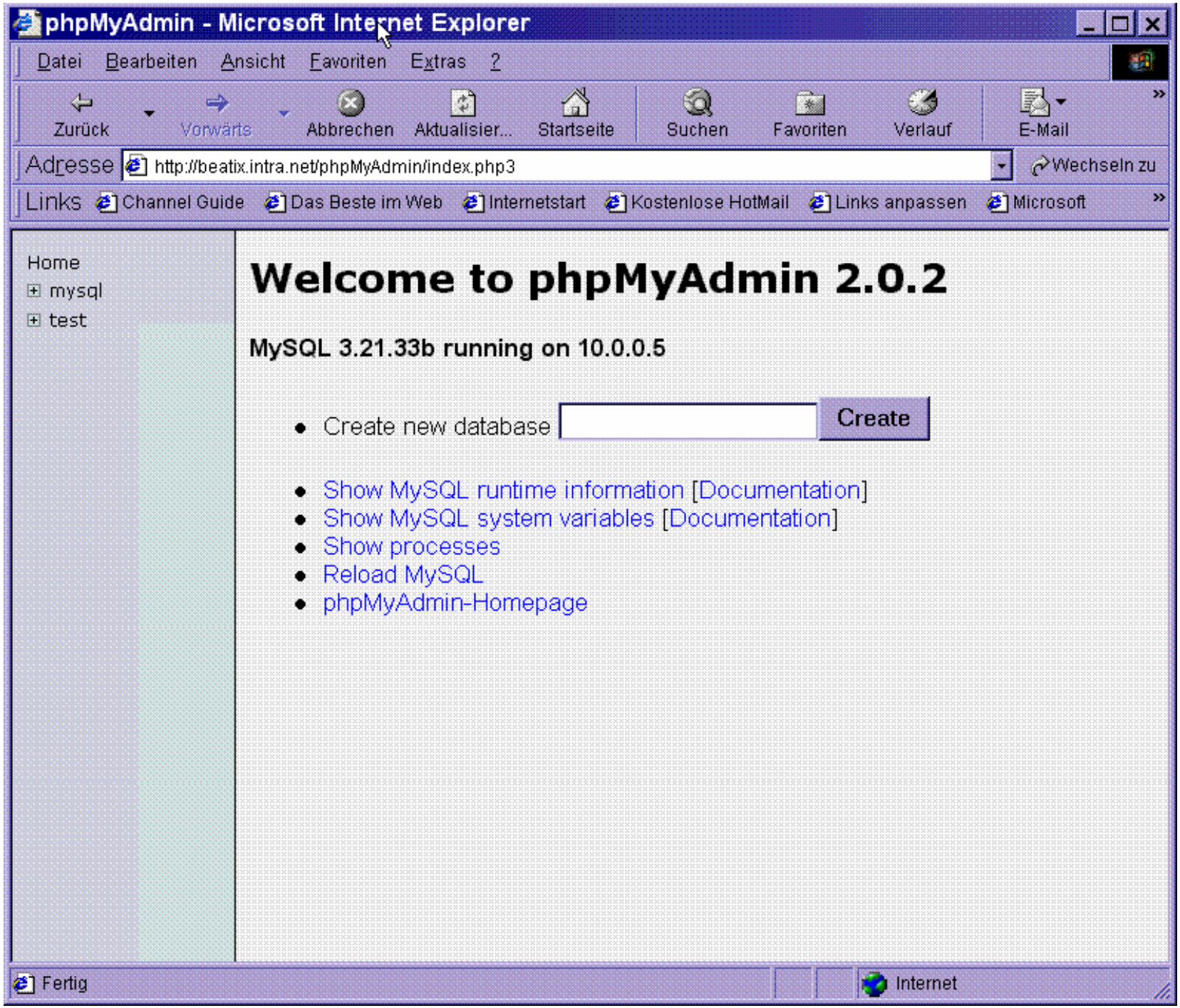


Abbildung:019

Schauen Sie sich nun die Systemvariablen an:

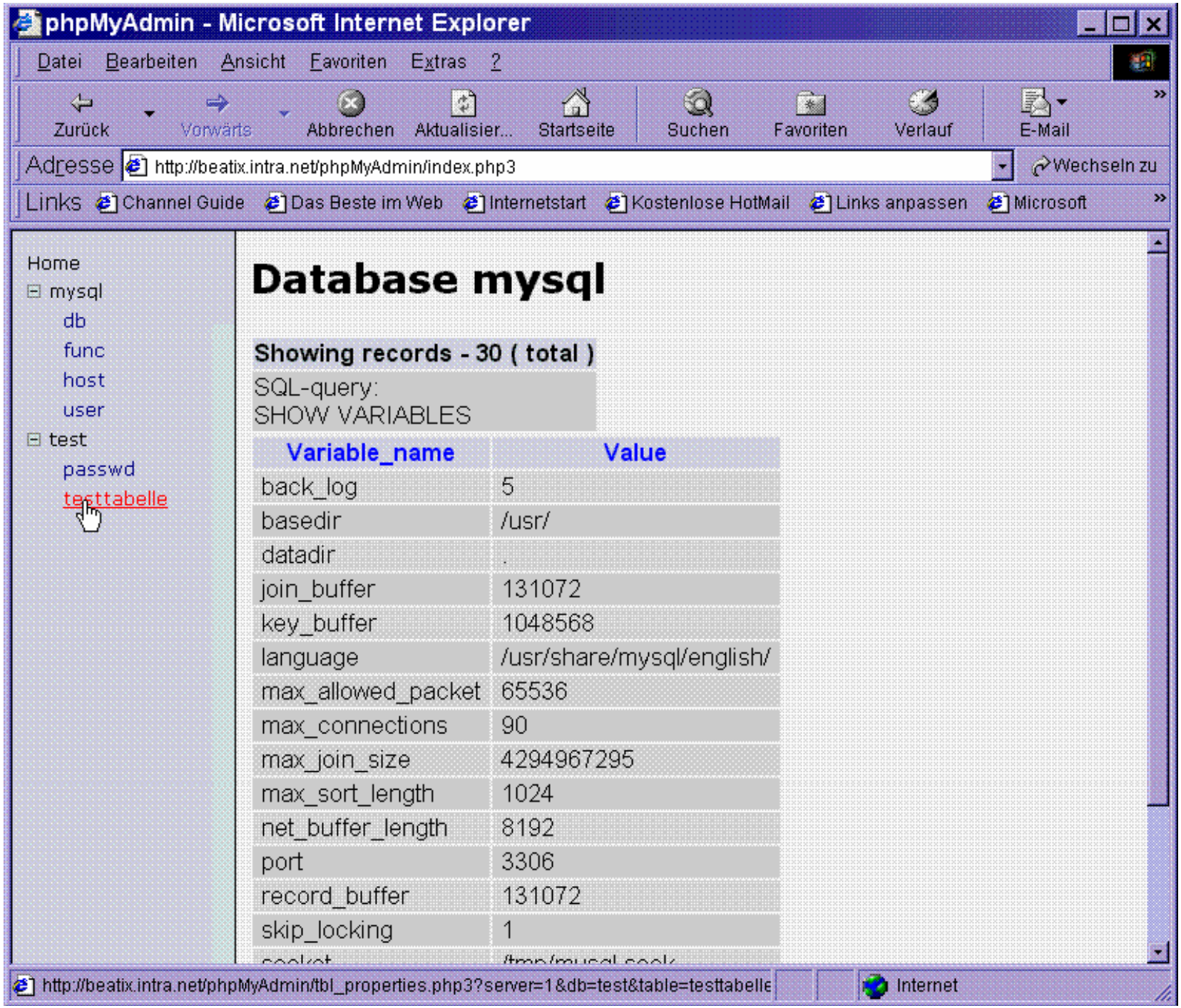
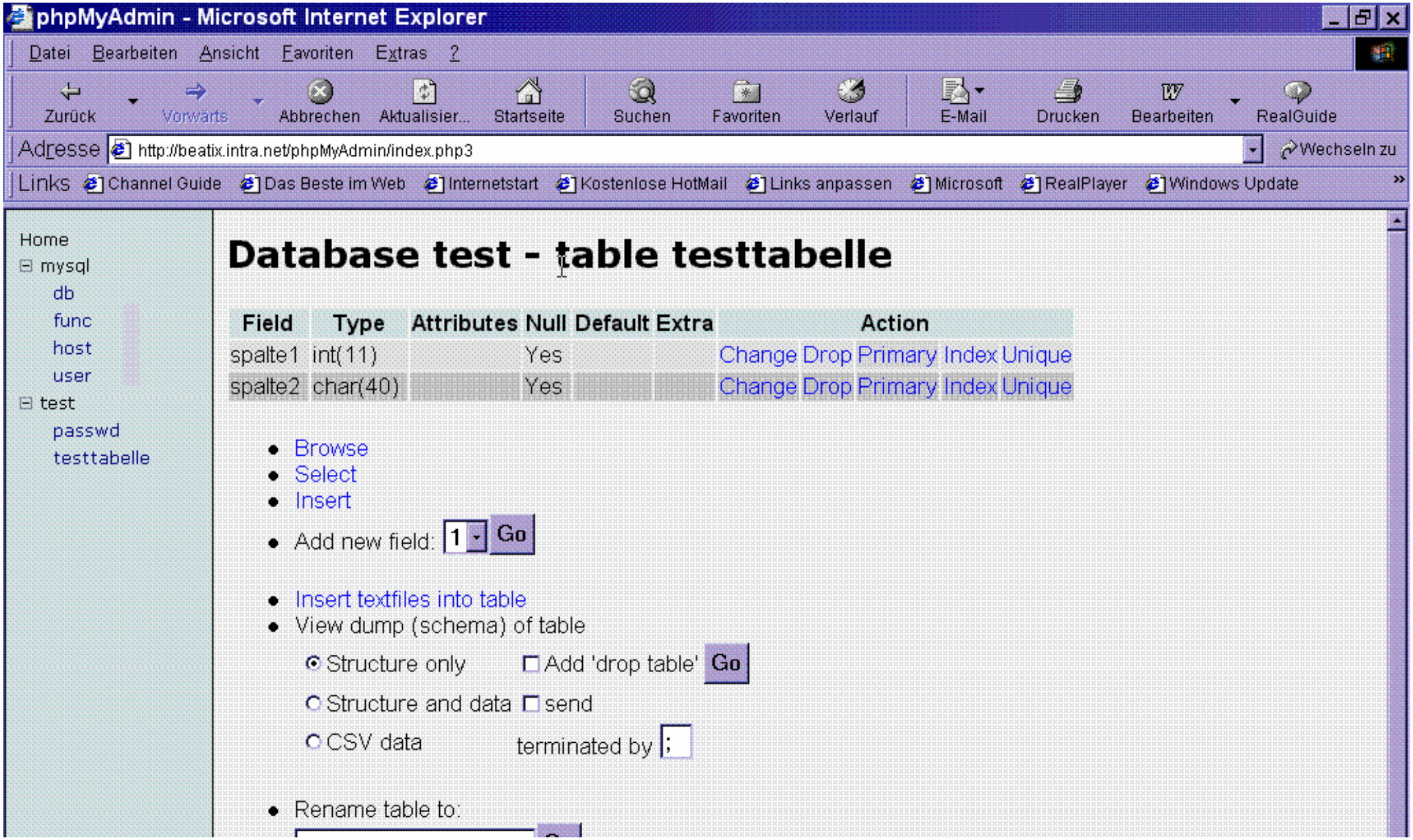


Abbildung:021

Einige sind für uns recht wichtig, z.B. **basedir**. MySQL betrachtet das Verzeichnis **/usr/** auf unsererm LINUX MySQL Server als Homeverzeichnis.

Die Variable **max_join_size** ist ein LIMIT, falls man 3 Tabellen mit 2000, 2000 und 1000 Einträgen verknüpfen möchte. (2000x2000x1000) Das Kartesische Produkt kommt hier zum Tragen, sehen Sie hierzu das Kapitel **Tuning**

Klicken Sie nun auf unsere **testtabelle**:



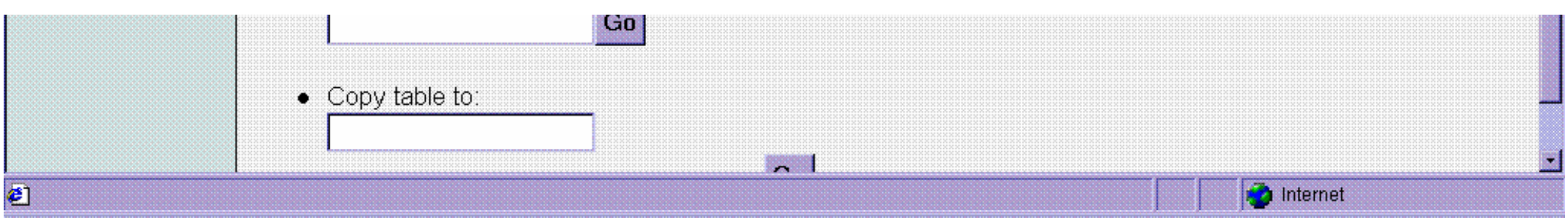
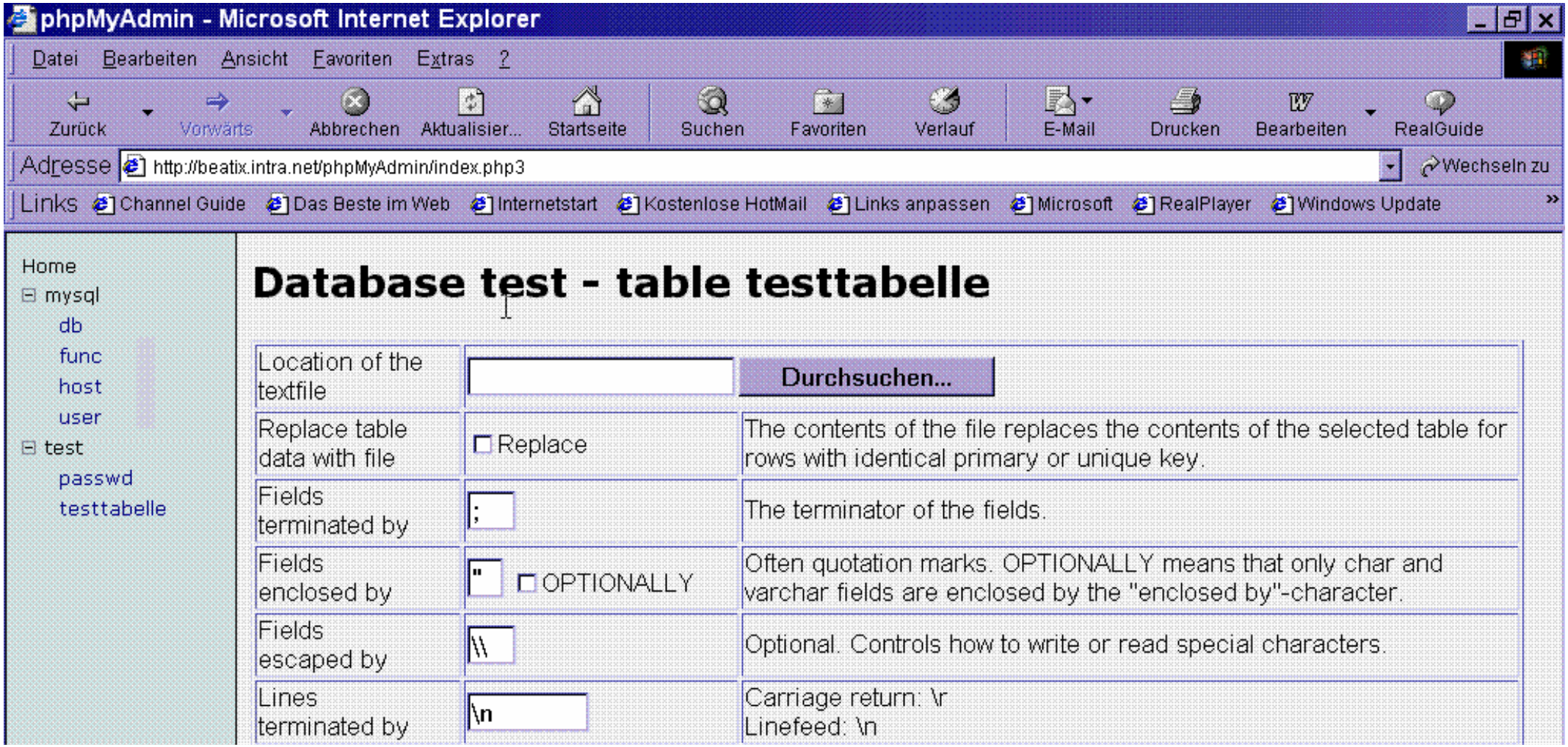


Abbildung:phpMyAdmin:Startup

Was Sie nun sehen, ist ein Administrationswerkzeug, welches Ihnen einfach per Menü alle Möglichkeiten von MySQL eröffnet. Sie können Datenbanken verändern, INDEXE und UNIQUE Felder anlegen, Attribute ändern, neue Spalten anlegen oder löschen, Datentypen verändern, Daten importieren u.s.w.

Hier ein Beispiel zum Importieren von Daten. Wie Sie nun sehen können, brauchen Sie sich nicht mehr mit der genauen Syntax von SQL auseinanderzusetzen, sondern nur noch anklicken, was Sie möchten, die zugehörigen Formulare ausfüllen, und das Kommando starten. Hier ein Beispiel zum Import von Daten:


phpMyAdmin - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Favoriten Extras ?
Zurück Vorwärts Abbrechen Aktualisier... Startseite Suchen Favoriten Verlauf E-Mail Drucken Bearbeiten RealGuide
Adresse http://beatix.intra.net/phpMyAdmin/index.php3
Wechseln zu
Links Channel Guide Das Beste im Web Internetstart Kostenlose HotMail Links anpassen Microsoft RealPlayer Windows Update
Home
mysql
db
func
host
user
test
passwd
testtabelle
Database test - table testtabelle
Location of the textfile
Durchsuchen...
Replace table data with file
Replace
The contents of the file replaces the contents of the selected table for rows with identical primary or unique key.
Fields terminated by
;
The terminator of the fields.
Fields enclosed by
"
OPTIONALLY
Often quotation marks. OPTIONALLY means that only char and varchar fields are enclosed by the "enclosed by"-character.
Fields escaped by
\\
Optional. Controls how to write or read special characters.
Lines terminated by
\n
Carriage return: \r
Linefeed: \n

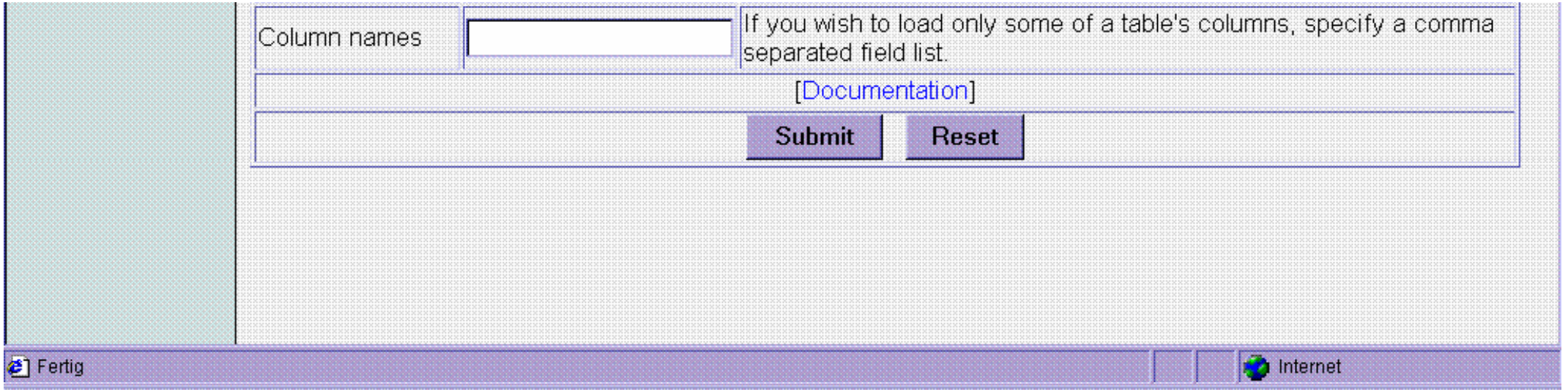


Abbildung:phpMyAdmin:Import von Daten

Lassen Sie sich die Inhalte der Datenbank ausgeben. Klicken Sie hierzu der Reihenfolge nach auf **testtabelle, browse** oder **testtabelle, select, go**. Drei Klicks, und Sie sehen den Inhalt der **teststabelle** Klicken Sie auf **spalte1**, und Sie sehen das SQL Statement eingeblendet, mit welchem die Spalteninhalte sortiert wurden.

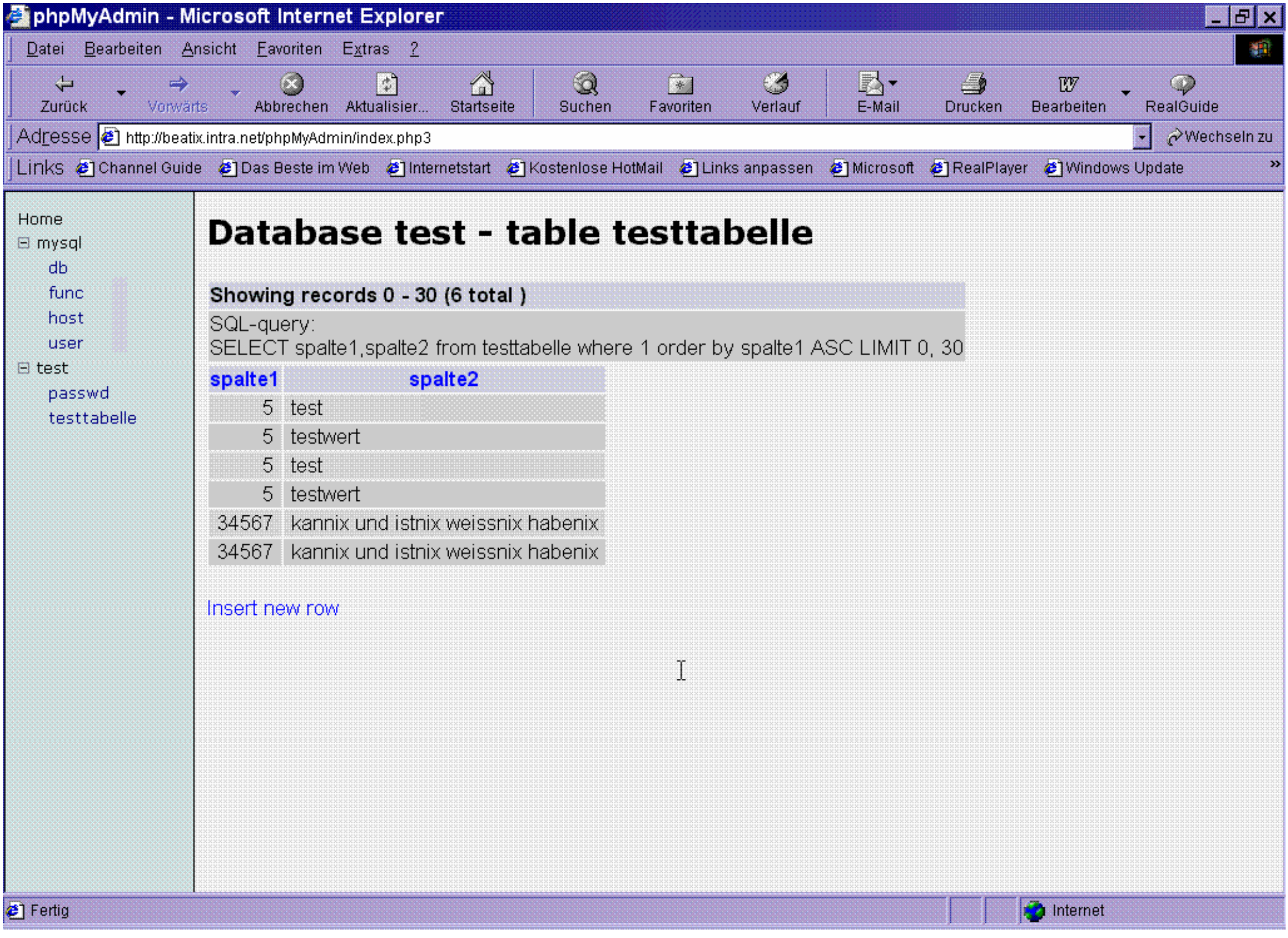

phpMyAdmin - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Favoriten Extras ?
Zurück Vorwärts Abbrechen Aktualisier... Startseite Suchen Favoriten Verlauf E-Mail Drucken Bearbeiten RealGuide
Adresse http://beatix.intra.net/phpMyAdmin/index.php3
Wechseln zu
Links Channel Guide Das Beste im Web Internetstart Kostenlose HotMail Links anpassen Microsoft RealPlayer Windows Update
Home
mysql
db
func
host
user
test
passwd
testtabelle
Database test - table testtabelle
Showing records 0 - 30 (6 total)
SQL-query:
SELECT spalte1,spalte2 from testtabelle where 1 order by spalte1 ASC LIMIT 0, 30
spalte1 spalte2
5 test
5 testwert
5 test
5 testwert
34567 kannix und istnix weissnix habenix
34567 kannix und istnix weissnix habenix
Insert new row
Fertig
Internet

Abbildung:phpMyAdmin:Testtabelle

Bei bestehenden Datenbanken könnte es manchmal von Interesse sein, zu wissen, welches **CREATE** Statement diese Tabelle erzeugt hat. Klicken Sie hierzu links auf **testtabelle**, klicken Sie unter **View dump** auf **Structure and data** und schließlich auf das **GO**, welches sich daneben befindet (2. **GO** von oben). Sie sehen nun die CREATE Struktur, mit der die Datenbank angelegt wurde, und die Inhalte der Datenbank, allerdings in INSERT Statements verpackt. Dieses Gebilde nennt sich DUMP:

```
# phpMyAdmin MySQL-Dump
#
# Host: 10.0.0.5 Database : test
# --------------------------------------------------------

#
# Table structure for table 'testtabelle'
#

CREATE TABLE testtabelle (
   spalte1 int(11),
   spalte2 char(45)
);

#
# Dumping data for table 'testtabelle'
#

INSERT INTO testtabelle VALUES( '5', 'test');
INSERT INTO testtabelle VALUES( '5', 'testwert');
INSERT INTO testtabelle VALUES( '34567', 'kannix und istnix weissnix
habenix');
INSERT INTO testtabelle VALUES( '5', 'test');
INSERT INTO testtabelle VALUES( '5', 'testwert');
INSERT INTO testtabelle VALUES( '34567', 'kannix und istnix weissnix
habenix');
```

Diese Informationen sollten Sie immer angeben, wenn Sie Ihre Probleme im Internet in den NEWSGROUPS oder in den Mailing-Listen schildern. Weil Ihre Probleme dann so einfach nachzuvollziehen sind, können Sie also erheblich schneller mit einer präzisen Antwort rechnen, die Ihr Problem löst. Der- oder diejenige, der diese Zeilen liest, kann dann mit Hilfe von DRAG und DROP Ihre Probleme auf der eigenen Datenbank einfach nachvollziehen, und Ihnen aufgrund direkt durch das Einfügen von kleinen Korrekturen in Ihren DUMP die Handreichung geben, die Sie als Einsteiger so dringend benötigen.

Beachten Sie nun, daß Sie in der Datenbank mysql ebenfalls alle Rechte haben, User und Userrechte zu der Tabelle testtabelle hinzuzufügen. Sie haben alle Rechte, inklusive der Rechte, die Datenbank zu löschen. Also Vorsicht.

Probieren Sie nun alles aus, was Sie seit Beginn dieses Tutorials umständlich mit Hilfe des MySQL Clients getestet haben. Legen Sie eine Tabelle **neutabelle** in einer neuen Datenbank **neudatenbank** an. Tragen Sie einen **neuuser** mit dem Paßwort **neupasswort** ein, der nur das Recht besitzt, Daten hinzuzufügen und Daten anzeigen zu lassen.

Sie können nun auch Daten von der eigenen Festplatte Ihrer Arbeitsstation, die Sie z.B. zuvor aus ACCESS exportiert haben, in die MySQL Datenbank auf Ihrem MySQL LINUX Server im Intranet oder Intranet übertragen. MySQL unterstützt erst seit der Version 3.22 den Befehl **LOAD DATA LOCAL INTO...**, mit welchem man Daten von der lokalen Festplatte in MySQL übertragen kann.

Mit dem Apache-Server und dem PHP3 Modul auf Ihrer Arbeitsstation können Sie nun mit Hilfe der MySQL Netzwerkfunktion in PHP3 die Daten überspielen:

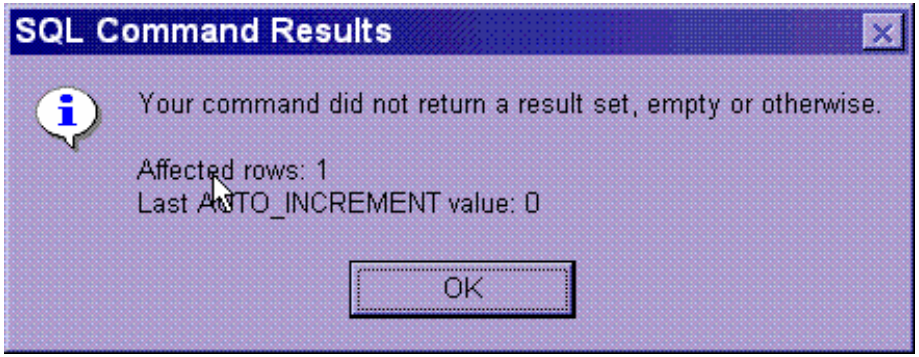

SQL Command Results
Your command did not return a result set, empty or otherwise.
Affected rows: 1
Last AUTO_INCREMENT value: 0
OK

Abbildung:Returnwert - MySQL

Dabei können Sie durch einfaches Anklicken des **replace** Buttons dafür sorgen, daß die Daten z.B. auf Ihrer MySQL Datenbank auf dem Internet-Server einzeln aktualisiert werden, sodaß der laufende Betrieb nicht gestört wird. Besser geht's nicht.




6.13 MySQL und JDBC Klasse 4

Damit Sie auch einmal ein JDBC Klasse 4 Interface betrachen und testen können, haben wir ein LIVE Beispiel ins Internet gestellt. Sie müssen jedoch JAVA in Ihrem Browser aktiviert haben. Klicken Sie nun auf http://www.rent-a-database.de/mysql/mysqllive.html und probieren Sie einige der o.a. Beispiele aus.







6.14 Test von PHP3 und MySQL

Mit folgendem Tutorial aus dem Kapitel Tutorial PHP 3 und MySQL können Sie schnell noch alle Einträge unserer **testtabelle** auf den Browser mit Hilfe eines PHP3 Sktiptes ausgeben lassen. Verwenden Sie dieses Skript als allgemeingültiges Skript zum Test Ihrer MySQL Datenbanken.







6.15 Update über ODBC Treiber

Eine erheblich elegantere Möglichkeit ist es, die Daten aus einer Microsoft Anwendung heraus über ODBC auf die Datenbank zu überspielen. Der Vorteil ist offensichtlich. Es werden nur diejenigen Daten über das Netzwerk übertragen, die sich auch verändert haben, alle anderen bleiben bestehen. Hierzu muß zunächst ein ODBC Treiber auf der Arbeitsstation installiert und eingerichtet werden. Folgen Sie hierzu den Anweisungen in Ihren Schulungsunterlagen. Einen aktuellen ODBC - Treiber laden Sie sich bitte von userem unserem Server: http://www.rent-a-database.de/support/mysql/odbc/. Hier finden Sie auch eine ausführliche Installationsanleitung mit der IP-Nummer unseres Schulungsservers im Internet.

Die Beispiel - Access - Datei laden Sie sich bitte aus dem Verzeichnis: http://www.rent-a-database.de/support/mysql/odbc/access.




7. Die Geschichte von SQL

SQL steht für Structured Query Language und wurde Ende der 70´er Jahre bei IBM in San Jose, Kalifornien als Abfragesprache für die relationale Datenbank DB2 entworfen. Es handelte sich hier ursprünglich um eine nichtprozedurale Sprache, die also keine Schleifen, Unterprogramme, Funktionen und Funktionsübergabeparameter u.s.w. enthält.

Hier ein Überblick nochmals über die Programmiersprachen der verschiedenen Generationen:

1. Generation: Maschinensprache. Diese stellt die unterste Ebene der Programmiersprachen dar. Sie besteht nur aus Zahlen, dementsprechend sind die Programme völlig unleserlich und schwer zu erstellen, jedoch laufen die Programme sehr schnell ab. Hier ein Beispiel der HEX-Notation: A900
2. Gereration: Assembler Sprachen. Diese orientieren sich gewöhnlich an den Maschinenbefehlen des Prozessors: **MOV AX,0** bedeutet dasselbe, wie A900, jedoch lassen sich diese Befehle einfacher merken. Komfortable Compiler (Übersetzer in Maschinensprache) beherrschten schon Schleifen, Sprungmarken und Unterprogramme. Damit wurde die Lesbarkeit sehr erhöht. Wer heutzutage Assembler z.B. unter LINUX kennen lernen möchte, der braucht einfach nur ein kleines C-Programm mit der Option **-S** übersetzen. Die GNU Assembler sowie alle GNU Compiler (C,C++, PASCAL, FORTRAN ....) übersetzen die Befehle einer Hochsprache zuerst in eine allgemeingültige Assemblersprache, bevor diese dann in die prozessorspezifische Assemblersprache und dann in Maschinensprache übersetzt wird. Dies hat die völlige Unabhängigkeit des Prozessors erst ermöglicht, wie man z.B. an LINUX sehen kann.
3. Generation: Höhere Programmiersprachen, wie z.B. BASIC, PASCAL, C. C++, ADA, Fortran... Diese sind schon an die menschlichen Denkweisen angepasst und ermöglichen, sofern man sich an einen Standard hält, den Austasch des Quellcodes zwischen verschiedenen Betriebssystemen und Prozessoren.
4. Generation: Nichtprozedurale Programmiersprachen (4GL) bescheiben nicht mehr, wie ein Problem gelöst werden soll, sondern nur noch welches speziele Problem gelöst werden soll. Diese Programmiersprachen sind natürlich nur für ganz spezielle Probleme konzipiert worden. Hierzu zählen SQL, NATURAL u.s.w.
5. Generation: KI Sprachen (Künstliche Intelligenz) erarbeiten Lösungen mit Hilfe von bestimmten Regeln, die definiert wurden. Hierzu sind ausgefeilte backtracking Mechanismen implementiert wurden, die bei der Suche nach Problemen, falls eine Sackgasse beschritten wurde, selbstständig aus diesen herausfinden können. Leider sind diese Sprachen noch etwas langsam. Hierzu zählen LISP (EMACS), PROLOG und SMALLTALK.

SQL ist also in dem Sinne keine Programmiersprache und dementsprechend einfach zu erlernen. Die SQL Befehle setzen sich aus zwei Teilen zusammen, der Data Definition Language (DDL) und der Data Manipulation Laguage (DML). Die DDL dient dem Aufsetzen der Datenbankstruktur, die DML dient der Manipulation der darin enthaltenen Daten.

Relationale Datenbanken arbeiten Mengen - und Gruppenorientiert. Wer sich also mit SQL beschäftigt, der muß sich ein wenig in Mengenlehre auskennen. Insbesondere das Studium der

INNER/OUTER/LEFT JOINS ist sehr wichtig, um Datenmengen zu handeln. SQL Datenbanken gehören alle zu den relationalen Datenbanken. Dr. E.F. Codd entwickelte 1970 Regeln, die eine relationale Datenbank definieren:

1. Ein relationales DBMS muß in der Lage sein, Datenbanken vollständig über seine relationalen Fähigkeiten zu verwalten.
2. Darstellung von Informationen: Alle Informationen in einer relationalen Datenbank (einschließlich Namen von Tabellen und Spalten) sind explizit als Werte in Tabellen darzustellen.
3. Zugriff auf Daten: Jeder Wert in einer relationalen Datenbank muß durch Kombination von Tabellenname, Primärschlüssel und Spaltenname eindeutig zu finden sein.
4. Behandlung von **Nullwerten**: Das DBMS behandelt Nullwerte durchgängig als unbekannte oder fehlende Daten und unterscheidet so von Standardwerten. Zahlen können also drei Werte annehmen, 0, einen Wert und NULL für "Wert nicht vorhanden". Man spricht hier auch von der **Dreiwertigkeit**.
5. Struktur einer Datenbank: Die Datenbank und ihre Inhalte werden in einem sogenannten Systemkatalog auf derselben logischen Ebene wie die Daten selbst - also in Tabellen - beschrieben. Demzufolge läßt sich der Katalog mit Hilfe der Datenbanksprache abfragen.
6. Abfragesprache: Zu einem relationalen System gehört mindestens eine Abfragesprache mit einem vollständigen Befehlssatz für Datendefinition, Manipulation, Integritätsregeln, Autorisierung und Transaktionen.
7. Aktualisieren von VIEWS (Sichten): Alle VIEWS, die theoretisch aktualisiert werden können, lassen sich auch vom System aktualisieren.
8. Abfragen und Bearbeiten ganzer Tabellen: Das DBMS unterstützt nicht nur Abfragen, sondern auch die Operationen für Einfügen, Aktualisieren und Löschen in Form ganzer Tabellen.
9. Physikalische Datenunabhängigkeit: Der logische Zugriff auf die Daten durch Anwendungen muß unabhängig von den physikalischen Zugriffsmethoden oder den Speicherstrukturen der Daten sein.
10. Logische Datenunabhängigkeit: Änderungen der Tabellenstrukturen dürfen keinen Einfluß auf die Logik der Anwendungen haben.
11. Unabhängigkeit der Integrität: Integritätsregeln müssen sich in der Datenbanksprache definieren lassen. Die Regeln müssen im Systemkatalog gespeichert werden. Es darf nicht möglich sein, die Regeln zu umgehen.
12. Verteilungsunabhängigkeit: Der logische Zugriff auf die Daten durch Anwendungen darf sich beim Übergang von einer nicht verteilten zu einer verteilten Datenbank nicht ändern.
13. Kein Unterlaufen der Abfragesprache: Integritätsregeln, die über die Datenbanksprache definiert sind, dürfen sich nicht mit Hilfe von Low-Level Sprachen umgehen lassen.

Dr. E. F. Codd hat hier in SQL alle Elemente der Algebra integriert, um Daten mengenmäßig zu erfassen, zu speichern, und diese in **Relation** zueinander zu setzen. Hierbei können Schnittmengen, Vereinigungsmengen, Restmengen u.s.w. gebildet und ausgegeben werden. Diese werden durch sogenannte JOINS, siehe Kapitel JOINS, durchgeführt.

Bei dem Aufbau von einer SQL Datenbank müssen die Daten so aufgeteilt werden, daß sie voneinander verschiedene, eindeutige Datensätze bilden. Dieser Prozeß wird in Kapitel Normalisierung beschrieben, und ist ein unentbehrlicher Schritt vor der eigentlichen Implementierung einer Datenbank. Hierzu werden, wie in Kapitel Entity Relationship Diagramme (ER) beschrieben,

ER - Diagramme erstellt. Ein solches Diagramm ist für die Entwurfsphase einer Datenbank unentbehrlich. Aus einem ER - Diagramm wird das Relationenmodell nach E.F. Codd entwickelt, wie in Kapitel Relationenmodell beschrieben. Eine saubere Vorarbeit erspart später viele Änderungen bei Datenformaten und Schnittstellen zur SQL-Datenbank.

Mit ANSI-92 SQL wurde ein gemeinsamer Standard geschaffen, an den sich PostgreSQL (für welches alle Beipiele hier im Handbuch ebenso gelten) z.B. hält. Leider gibt es hier verschiede Ebenen des Sprachunmfages: Entry, Intermediate und Full. Zur Erfüllung des vollen Sprachumfanges müssen folgende Merkmale erfüllt sein:

- Anbindung an andere Datenbanken über z.B. ODBC
- Rollbare Cursor dienen dem Scrolling über Fehler und Datensätze hinweg
- Dynamisches SQL bietet die Möglichkeit, SQL-Befehle vor der Ausführung vorzubereiten
- Verschiedene JOIN - Formen

Im Grunde erfüllen nur wenige SQL - Datenbanken den Standard "full". Dennoch kann man auch mit diesen hervorragend arbeiten, da es viele Umsschreibungen gibt, wie in Kapitel Workarounds beschrieben ist. In der Praxis hat jede Datenbank ihre Eigenheiten, sodaß oftmals der Wechsel zwischen den Datenbanken nicht einfach möglich ist.

Mit der Abfragesprache SQL können folgende Operationen mit den Daten ausgeführt werden:

- Veränderung der Struktur einer Datenbank
- Zugriffsrechte auf Strukturen vergeben
- Informationen auslesen
- Inhalte verändern

Die Sprache SQL ist im Laufe der Zeit um einige Kategorien von besonderen Funktionen erweitert worden, von denen man fast alle auch in **MySQL** wiederfinden:

- **Aggregatfunktionen** (COUNT, SUM, AVG, MAX, MIN, VARIANCE, STDDEV...) sind im ANSI Standard definiert und in vielen SQL Datenbanken erheblich erweitert worden.
- Datum - und Zeitfunktionen
- Arithmetische Funktionen (ABS, SIN, COS, LN, LOG, MOD, FLOOR, CEIL,...) sind in **MySQL** in einem sehr großen Umfang implementiert
- Zeichenfunktionen (TOLOWER, TOUPPER, LPAD, RPAD, REPLACE, SUBSTR, INSTR..) sind in MySQL ebenfalls in großem Umfang implementiert
- Umwandlungsfunktionen (TO_CHAR, TO_ASCII ...)
- Andere Funktionen (GREATEST, LEAST...)

Als Klauseln (clauses) bezeichnet man bei SQL z.B.:

- WHERE
- STARTING WITH
- ORDER BY
- GROUP BY
- HAVING

Diese Klauseln sind alle Optionen des Befehls **SELECT**, dessen vielen Parameter in Kapitel SELECT beschrieben ist, und dienen dazu, Mengen zu beschränken, zu sortieren und deren Ausgabe zu ordnen.

Das praktische an SQL ist, daß wenn man die Abfragesprache einmal erlernt hat (so schwer ist es wirklich nicht), kann man aus riesigen Datenmengen wirklich alle Informationen auslesen, die man braucht. Z.B. kann man Daten sortieren, verändern, filtern, Statistiken erheben, u.s.w. Mit Hilfe von Datenbankschnittstellen, wie z.B. ODBC und ASCII Im/Export kann man Daten mit beliebigen Datenbanken austauschen und sogar diese mit einbinden. Die ODBC Schnittstelle, genauer in Kapitel ODBC - Schnittstelle beschrieben, dient hier als **Datenbank unabhängige** Schnittstelle zu Applikationen unter Windows und UNIX.





8. MySQL im praktischen Einsatz - Beispiele

MySQL wird für fast alles verwendet. Dank frei verfügbaren Quellcodes sind Einsätze möglich, für die andere professionelle Datenbanken nicht in Frage kommen:

1. **Live Datenbank mit JDBC Interface** zum Erlernen der SQL Sprache: http://www.rent-a-database.de/mysql/mysqllive.html
2. **Verteilte Datenbanken durch E-Mail Synchronisation** ist die Möglichkeit, Filialen nur über eine lokale DIAL-ON-DEMAND Leitung international miteinander zu verbinden. Siehe hierzu Kapitel Verteilte Datenbanken über Mail
3. **MySQL und BIND8**, dem UNIX DNS-Server. Hier hat Mitchel Broom den DNS-Server einfach mit der MySQL Datenbank verknüpft. Der Vorteil: Man kann einfach Statistiken erstellen, Kundendaten auswerten, eine Anbindung an die Finanzbuchhalung zur Abrechnung der DNS-Kosten schaffen, u.s.w. Siehe hierzu auch http://bind.linuxos.net/table.php3
4. Passwortverwaltung (MOD_AUTH) in Zusammenhang mit dem APACHE WWW-Server. Hier werden Mitglieder in MySQL erfasst, die dann bestimmte Rechte zu Unterverzeichnissen des WWW-Servers erhalten (oder auch nicht) können.
5. **Echtzeiterfassung von Daten** zur Qualitätskontrolle. Fa. BASF und Bayer setzen MySQL zur Echtzeitauswertung und Qualitätsmanagement bei chemischen Prozessen ein. Ein LINUX Server mit einer 256'er seriellen Karte und MySQL liefert kontinuierlich statistische Auswertungen für die Messwarte. Es wurden mehrere DEC-VAX'en, eine DEC ULTRIX mit einem Fortranprogramm zur Aufarbeitung der Daten, ein SQL-Server, ein PC mit ACCESS zur Erstellung der Statistiken durch einen Pentium 75 mit LINUX, 256'er serieller Karte, MySQL und ein paar JAVA-Applets ersetzt. Beipiele finden Sie im Kapitel JDBC Treiber
6. **Echtzeiterfassung und Auswertung von Log-Daten**. Unter UNIX werden eine Vielzahl von Logdaten in Dateien geschrieben. Diese können in MySQL kontinuierlich über eine Named Pipe mit Buffer eingelesen und statistitisch in Echtzeit ausgewertet werden.
7. **Trouble Ticket Systeme mit MySQL** Zur Organisation von Fehlermeldungen werden in größeren Firmen teuere Trouble Ticket Systeme eingesetzt. Es gibt inzwischen professionelle, in PHP 3 geschriebene Software, die die Daten über MySQL verwaltet. Es lassen sich einfach Anschlüsse an z.B. SAGE KHK Software für die Rechnungsstellung realisieren.
8. **Einkaufs-SHOPS** im Internet für MySQL gibt es inzwischen zahlreiche in PERL oder PHP3. Der eindeutige Vorteil ist die Möglichkeit zum Anschluß an das hauseigene Warenwirtschaftssystem (SAP, SAGE KHK). Hierzu werden die SQL-Datenbanken über eine Firewall (Siehe auch http://www.little-idiot.de/firewall/ verbunden, oder über einen Arbeitsplatz-PC, der an das Internet angebunden ist, die Daten über ODBC eingelesen, überprüft, und dann in das Warenwirtschaftssystem überspielt.
9. **SQL Server Cluster** lassen sich mit MySQL entgegen der landläufigen Meinung einfach realisieren. Hierzu lesen Sie bitte Kapitel Verteilte Datenbanken, wo sich auch die komplette Software zum Spiegeln der MySQL Datenbanken in Echtzeit findet.
10. **Bilddatenbanken** Für Designer, Grafiker und Agenturen dürfte folgende Bilddatenbank sehr

interessant sein, die sich hier findet: http://www.riddleware.com/php3/album/index.php3. Die Bilder lagern auf der Festplatte in irgendwelchen Verzeichnissen, völlig ungeordnet und werden dann automatisch, bevor sie an den Browser ausgeliefert werden, verkleinert (echte Verkleinerung der Bildgröße). Das Projekt wurde mit APACHE, PHP3, PERL, MOD_PERL ImageMagik und natürlich MySQL realisiert. Auf dieser Site finden wich viele, weitere Beispiele.

11. **Kalender** zur Koordination von Terminen in größeren Gruppen. Mit MySQL wurden unzählige Kalender realisiert, die alle frei im Internet verfügbar sind. Inzwischen gibt es auf jeder Homepage von Suchmaschinen (Yahoo, Lycos) Kalender, die unter MySQL und PHP3 bzw. PERL programmiert wurden.
12. **Dynamische Websites** mit MySQL gehört zu den einfachsten Dingen die sich mit MySQL und dem Apache WWW-Server lösen lassen. Dank der Geschwindigkeit von MySQL ist diese Datenbank die Nummer 1 unter den SQL Datenbanken, mit denen riesige namhafte Websites betrieben werden.
13. **Dezentrale Datenerfassung** mit einer MySQL Datenbank im Internet über Browser ist inzwischen schon weit verbreitet. Als Frontend arbeitet hier oft Microsoft ACCESS.
14. **Bookmark Verwaltungsprogramme**, wie z.B. Denkwerk sie anbietet, finden sich im Quellcode unter http://www.freshmeat.net im PHP - Quellcode. Basis ist immer MySQL.








9. Grundwissen zum Entwurf einer Datenbank

Abgesehen von der richtigen Dimensionierung der Hardware, die sich heutzutage nicht mehr stellt, da heutige PC´s unter LINUX schon das leisten, was nur wenige Rechenzentren vor ca. 2 Jahren leisten konnten, stellt sich die Frage nach der Planung der Datenbankstruktur, auf die in Kapitel Planung der Datenbankstruktur genau eingegangen wird. Wie wichtig die Datenbankstruktur ist, wird im Kapitel SQL Tuning deutlich, wo eine falsche Struktur der Datenbank (INDEX Felder, JOINS) die Geschwindigkeit drastisch (Faktor 10-1000 ist möglich) verlangsamt.







10. Was ist MySQL ?

MySQL ist eine echte Multi-User, Multi-Treaded SQL Datenbank und wird von allen großen Providern oder auch Suchmaschinenbetreibern eingesetzt. MySQL ist eine CLient/Server Implentierung, die aus einem Server-Dämon **mysqld** und vielen Client Programmen, sowie Bibliotheken für PERL, PHP/3, PHP/4 sowie ASP besteht.

SQL ist eine standardisierte Datenbanksprache, die das Speichern, Updaten und den Zugriff auf Informationen erleichtert. Beispielsweise kann man Produktinformationen eines Kunden auf einem WWW-Server speichern und abrufen. **MySQL** ist äußerst schnell und flexibel genug, um sogar Bilder und Log-Dateien darin abzulegen. In der Praxis ist MySQL sehr viel schneller, als z.B. ORACLE oder INFORMIX.

Die wichtigsten Eigenschaften von **MySQL** sind Geschwindigkeit, Stabilität und einfache Bedienbarkeit. MySQL wurde ursprünglich entwickelt, weil auf TCX (dem Server der Entwickler) ein SQL Server benötigt wurde, der sehr große Datenbanken handeln konnte, und zwar um eine Größenordnung schneller, als die Datenbankhersteller damals liefern konnten. **MySQL** ist nun nun seit 1996 auf vielen Tausend Sites im Internet und Intranet im Einsatz und erfreut sich hier wachsender Beliebtkeit. **MySQL** ist bei vielen tausend Unternehmen im täglichen Einsatz, von denen über 500 mehr als 7 Millionen Einträge bzw mehr als **100 Gigabyte** an Daten mangagen. Auf der Site http://www.rent-a-database.de ist eine Auszug einer Referenzliste zu finden.

MySQL ist geradezu prädestiniert zur Beschleunigung bestehender ACCESS Datenbanken. Diese Kombination ermöglicht es, mit nur kleinen Änderungen in ACCESS die Datenbankabfragen um Faktor 3-100 zu beschleunigen.

Der Grundstock, um den **MySQL** herum gebaut worden ist, ist eine Liste von Routinen, die sich im täglichen Einsatz seit Jahren bewährt haben. Obwohl **MySQL** dauernd weiter entwickelt wird, hat es zu jedem Zeitpunkt der Entwicklung stets zuverlässig und stabil gearbeitet.







11. Das Handbuch und die Autoren

Das Handbuch zu MySQL ist im Original von David Axmark, Michael Widenius, Paul DuBois und Kim Aldale geschrieben worden. Die deutsche Übersetzung und viele ergänzende Kapitel wurde von G. Stepken, August 1999 geschrieben.







12. Planung der Datenbankstruktur

Im folgenden wird anhand eines prakischen Beispiels die Umsetzung von der Idee bis zur fertigen Datenbank beschrieben. Typische Fehler bei der Planung und Umsetzung der Implementierung einer relationalen Datenbank unter MySQL werden dargestellt und aufgekärt. Wer sich unnötig mit theoretischem Ballast zunebeln möchte, dem sei das Standardwerk von Gottfried Vossen: Datenbankmodelle, Datenbanksprachen und Datenbank-Management-Systeme empfohlen. Zum Verständnis der in der Datenbanktechnik gebräuchlichen Sprache sei das Lexikon der Datenbanktechnik am Ende den Handbuches emfohlen.







12.1 Vorüberlegungen

Zu einer Datenbank gehört immer Paperware als Dokumentation dazu. Man kann zwar spontan Datenbanken implementieren, jedoch passieren auch Profis Fehler bei der Planung der Struktur, die dann eventuell langwierige Änderungen im Quellcode der WWW-Interfaces oder der Datenbankstruktur nach sich ziehen. Besonders unangenehm wird es, wenn bereits die Datenbank genutzt wird, und dann aber dringende Änderungen der Struktur notwendig sind. Es müssen dann ja auch alle Frontends gleichzeitig ausgetauscht werden, was dann sicher zum Stillstand des Unternehmens führen würde. Wer ACCESS als Frontend einsetzt, der wird bei vielen Clients dann sicher große Probleme haben. Diese dürften Datenintegrität, Funktionalität und die Konvertierung der unterschiedlichen Datenbankstrukturen ineinander betreffen. Es lohn sich also stets, schon vorher mögliche Änderungen der Struktur mit einzuplanen.







12.2 Entity Relationship (ER) - Modell

Das ERM bzw. das ERD (Diagramm) gehört zu den wichtigsten Modellen, mit denen Datenbankstrukturen entworfen werden. Aus den Entity - Relationship Modellen können nicht nur die Strukturen von SQL Datenbanken, sondern auch die Strukturen von allen anderen Datenbanken erstellt werden. Hierzu geht man nach einem Frage - Antwortschema vor. Bevor wir nun diese Strukturen erstellen, müssen unbedingt noch die Fachtermini geklärt werden:




12.3 Begriffe aus der Datenbanktechnik

- Eine **Entity** ist immer ein existierendes Objekt, eine Beobachtung, eine Transaktion oder Person, über welches Informationen in einer Datenbank abgelegt werden. Beispiel:

  Eine bestimmte Person (Frau Meier), ein realer Gegenstand (Das Buch "Vom Winde verweht"), oder einfach nur ein Ereignis, wie **Kaffe trinken** oder **Buch ausleihen**.
- Eine **Entitätsmenge** (Entity - Set) ist die Zusammenfassung einzelner Entities, die ähnliche oder vergleichbare Eigenschaften haben. Beispiel: Alle Bücher in einer Bibliothek, die Einwohner einer Stadt oder alle Raucher.

  Entitätsmengen (wir erinnern uns an die Mengenlehre, der ja SQL zugrunde liegt) können sich natürlich mit der Zeit (Merke: Zeitabhängigkeit) ändern.
- Attribute sind Eigenschaften oder Merkmale einer Entität. Sie sind die typischen Eigenschaften einer Entität und charakterisieren diese Entität. Beispielsweise sind die Kunden einer Firma eine Entitätsmenge, also eine Ansammlung von einzelnen Entitäten. Sowohl eine Entity als auch eine Entitätsmenge werden durch Attribute charakterisiert. Als Attribute werden z.B. Kundennummer, Name, Vorname, Wohnort oder Straße bezeichnet. Es gibt sowohl einwertige Attribute zusammengesetzte Attribute. Ein Buch (also eine Entity) kann mahrere Autoren haben, ist also eine Entity mit mehrwertigem Attribut. Ein typisches Beispiel für ein zusammengesetzes Attribut ist die Adresse, die aus Postleitzahl, Ort und Straße besteht.
- Ein **Entitätstyp** beschreibt eine Menge von Entitäten mit gleichen Attributen. Ein Entitätstyp könnte ein Buch sein, welches mit den Attributen: Titel, Autor, Verlag, ISBN-Nummer in der Datenbank definiert wurde. Der Entitätstyp darf nicht mit Entitätsmenge verwechselt werden ! Eine Entitätsmenge ist eine Sammlung von vielen Büchern mit bestimmten Gemeinsamkeiten (alle eines Autors), ein Entitätstyp ist mehr als nur die Bücher eines Autors, es sind alle Bücher, die einen Autor, einen Verlag, eine ISBN-Nummer haben. Damit lassen sich alle im Buchhandel käuflichen Bücher beschreiben, aber nicht die privat gedruckten (wie z.B. dieses Handbuch für MySQL hier). Der Grund liegt darin, daß dieses Handbuch keine ISBN-Nummer besitzt. Es ist also nicht vom gleichen Entitätstyp, sehr wohl kann es aber zu der Entitätsmenge der Bücher einer Bibliothek gehören. Jede Entität eines Entitätstyps kann über die Attributwerte eindeutig identifiziert werden. In der Praxis genügt aber oft schon die Angabe einiger Attribute, um die Entität eindeutig bestimmen zu können. Diese Attribute nennt man Schlüsselattribute. und deren Zusammenfassung die Schlüssel.
- Schlüssel (key) oder Schlüsselkandidaten (candidate key) ist die minimale Kombination von Attributen, die eine Entität aus einer Entitätsmenge eindeutig identifiziert. Minimal bedeutet hier, daß man nicht alle Attribute angeben muß, damit man eine Person oder ein Buch eindeutig identifizieren kann. Die Zahl der notwendigen oder auch hinreichenden Attribute hängt natürlich stark von den Inhalten ab. Möchte man z.B. eine Person eindeutig identifizieren, so muß man sich die Attribute einer Person (also Entity) anschauen:

  ```
  PERSON, beschrieben durch PersNr, Name, Vorname, Adresse, Geburtsdatum, Beruf
  ```

  Leider ist die Kombination von den Eigenschaften (Attributen) dieser Entity PERSON nicht minimal, ist also kein Schlüssel. In der Praxis ist das auch wenig praktisch, da man, um eine Person aufzufinden, oder mit anderen Enties zu verknüpfen (trinkt Kaffee), man ja alle Attribute in die Datenbank eintippen muß, damit eine Abfrage erfolgreich sein kann. Besser ist es hier, das naheliegenste als eindeutigen Schlüssel zu bestimmen, die PersNr. Wie man das aus der Praxis mit Personalausweisenummern kennt, ist diese Nummer immer ein eindeutiger Schlüssel für Personen. Ein Schlüssel wäre aber auch die Kombination von Name, Vorname und Geburtsdatum. Krankenkassen z.B. speichern Personen stets so ab. Über die Angabe dieser drei Attribute finden die Mitarbeiter jedes Mitglied und alle Vorgänge über dieses Mitglied wieder. Das diese Kombination ein eindeutiger Schlüssel ist, liegt an der statistischen Verteilung der Daten. Bei der Angabe von Thomas Schmitz und einem Datum könnte es Schwierigkeiten geben, bei Zwilligen jedoch nicht, da diese von ihren Eltern garantiert nicht auf den gleichen Namen getauft werden. Falls der Schlüssel also nicht in jedem Falle eindeutg ist, macht das auch nichts. Die Datenbank gibt dann ein paar wenige zur Auswahl aus. Trotzdem wird diese Kombination als eindeutig bezeichnet.
- Primärschlüssel (primary key) ist häufig identisch mit einem Schlüssel, z.B:

  ```
  PERSON (PersNr, Name, Vorname, Adresse, Geburtsdatum, Beruf)
  ```

  Wenn man sich zwischen mehreren eindeutigen Schlüsseln oder auch zusammengesetzen Schlüsseln entscheiden kann,

nimmt man immer den einfachsten Schlüssel, der eine Entität (PERSON) eindeutig beschreibt. Dies ist heir die PersNr, also die Personalnummer. Es häfig eine laufende Nummer (Siehe hier Kapitel CREATE unter "autoincrement").

So, nun sind vorläufig alle wichtigen Begriffe der Datenbanktechnik erklärt worden. Man muß sich also viele Gedanken darüber machen, welche Attribute zu welcher Entität gehören, welche veränderlich sind, welche von anderen abhängig sind (Adresse und Postleitzahl), und wie man diese am besten so ordnet, daß z.B. Redundanzen vermieden werden. Wir möchten ja nicht unsere Datenbank unnötig aufblähen.








12.4 Ziele bei der Struktur der Datenbank

Der Ansatz der relationalen Datenbank hat einige Ziele, die man stets erreichen sollte. Hierzu wird z.B. das Verfahren der Normalisierung von E.F. Codd eingesetzt, hierzu aber mehr später. Fassen wir die wichtigsten Ziele der Einführung einer Datenbank zusammen:

1. Datenintegrität, darunter fallen auch wichtige Themen, wie Datenkoinsistenz, Datenschutz und Datensicherung, alles Themen, die man schon beim Entwurf der Datenbank berücksichtigen sollte, z.B. durch die Einführung von weiteren Attributen (Datum, Uhrzeit der Änderungen, zusätzliche Attribute, die Replikationen ermöglichen, u.s.w.
2. Redundanzfreiheit, darunter versteht man, daß man nicht unnötig Daten mehrfach in der Datenbank ablegen muß, was die Größe und damit die Geschwindigkeit negativ beeinflußt.
3. Datenunabhängigkeit ist die Aufsplittung aller Attribute und die Einordnung in eigene Spalten. Siehe hierzu auch die erste Normalform von Codd.
4. Benutzerorientierte Sicht der Daten. Hiermit ist gemeint, daß man eine Person zuerst meist nach dem Namen fragt, und wahrscheinlich, falls der Schlüssel nicht eindeutig ist, nach dem Wohnort, Adresse und zu allerletzt wohl nach seinen Hobbies oder sonstigen Neigungen.
5. Unterstützung der Datenmanipulation. Hiermit ist der Komfort des Benutzerinterface gemeint. Das Blättern mit dem Cursor durch die Datenbank, die einem vielleicht von ACCESS her völlig normal vorkommt, ist unter SQL allgemein nicht so einfach zu realisieren. Die Sprache HTML zur Abfrage von Datensätzen erlaubt kein Scrolling durch die Datensätze, hierfür muß man auf JAVA (JDBC) oder proprietäre C - Kompilate zurückgreifen. Das einfachste ist jedoch, ACCESS weiterhin als Frontend zusammen mit **MySQL**einzusetzen.
6. Koordinierung des Multi-User Betriebes. hier sind verschiedenste Probleme zu lösen, z.B. Locks, Server - Replikation, Server-Spiegelung, Backups, Synchronisation von verteilten Datenbanken.

Viele dieser Themen werden in einzelnen Kapiteln in diesem Handbuch intensiver beleuchtet. MySQL läßt Sie bei keinem dieser Probleme im Stich, auch wenn manchmal ein Workarounds notwendig ist. Alle Datenbanken sind aber mit Problemen behaftet, also nur ruhig Blut. LINUX und MySQL gehören zu den am ehrlichsten und besten dokumentierten Softwarepakteten überhaupt - es ist nämlich alles sogar im Quellcode verfügbar....







13. Copyright des MySQL Handbuches

Das Copyright liegt bei G. Stepken. Der kommerzielle Druck und Verkauf dieses Handbuches, komplett oder auch ausschnittsweise, sowie das Kopieren auf Datenräger, wie Diskette oder CDROM für kommerzielle Zwecke ist untersagt. Der Ausdruck sowie das Kopieren ausschließlich zu privaten Zwecken hingegen ist erlaubt. Das Setzen von Links auf dieses Handbuch im Internet, das Framen sowie Mirroring ist ausdrücklich erwünscht.

Wer die Tutorials zu Schulungszwecken verwenden möchte, der wird um eine kleine finanzielle Spende zugünsten der Weiterentwicklung der Tutorials bzw. von MySQL selber gebeten. Spenden sind natürlich auch so willkommen.







14. Die Historie von MySQL

MySQL stammt von **mSQL** ab, mit dem Zweck, die Tabellen mit eigenen, schnellen ISAM Routinen zu verbinden. Nach einigen Testläufen wurde festgestellt, daß **mSQL** nicht schnell und flexibel genug war. Das Ergebnis war, daß **MySQL** völlig neu geschrieben wurde, allerdings mit demselben API Interface, wie **mSQL**.







15. Support für MySQL und Programmierschnittstellen

Für MySQL hat sich inzwischen, ebenso wie für **PostgreSQL**, ein unglaublicher Fundus an fertigen Lösungen zusammengefunden. Fast alle Arten von Anwendungen sind dabei. Für fast alle Programmiersprachen und Betriebssysteme gibt es nun Entwicklungs - Toolkits, die allerdings unterschiedlich gut gepflegt und supportet werden. PHP3, PERL, Embedded PERL, ePERL, JAVA und die vom Entwicklungsteam bereitgestellten C++ API´s sind sicher erste Wahl. Weniger gut supportet sind hingegen ausgefallene Programmiersprachen und einige Betriebssysteme, wie z.B. Apple Mac. Wer unter Microsoft MySQL ansteuern möchte, dem stehen durch den ODBC Treiber (myodbc) alle Möglichkeiten offen, unter einer beliebigen Programmiersprache (Visual Basic, Visual C++, JAVA, WSH ...) Anwendungen zu programmieren. Microsoft ODBC gehört aber sicher zu den langsamsten und schlechtesten Lösungen, auch wenn es ein Standard unter den Datenbankschnittstellen ist.







15.1 Die Mailing-Listen Beiträge der letzten 2 Jahre als HTML-Archiv

Glücklicherweise gibt es die "dummen" Fragen der letzten 2 Jahre zusammengefasst als Mail-Archiv auf den Sites http://www.bitmechanic.com/mail-archives/mysql/ oder http://www.tryc.on.ca/archives/mysql/ Wer also Fragen hat, der möchte sich bitte immer zuerst an das Bitmechanic oder TRYC - Archiv wenden. Zu 99.99% findet man dort alle Herzensfragen beantwortet.




15.2 Erstellen eines BUG-Reports

Support für MySQL wird vielerorts angeboten. bevor Sie sich jedoch an die Entwickler oder an kommerzielle Supportstellen wenden, lesen Sie bitte ausführlich das Handbuch durch. Viele Probleme lösen sich hierbei von alleine. Trotzdem können natürlich BUGS in **MySQL** enthalten sein. Um einen Bug-Report zu erstellen, sollte man das Formular **mysqlbug** aus dem Verzeichnis ../scripts (bei der Source-Distribution) bzw. aus dem Verzechnis ../bin (bei der Binary-Distribution) benutzen. Dieses Script fragt nach weiteren Informationen bezüglich des verwendeten Betriebssystems, der Konfiguration, u.s.w., welche unentbehrlich bei der Rekonstruktion des Fehlers sind. Ohne diese ausführlichen Informationen kann Support generell nicht geleistet werden. In dem BUG-Report müssen unbedingt folgende Informationen enthalten sein:

1. Versionsnummer von MySQL (Siehe mysqladmin version.mysqladmin),
2. Distributor Ihrer MySQL Version
3. Hardwarehersteller
4. Betriebssystemversion (UNIX: shell> uname -a)
5. Speicher (Festplatte und RAM)
6. Wenn es bei der Kompilierung Probleme gibt, die Fehlermeldungen des Kompilers, Das "configure" Script und das "Makefile"
7. Die Ausgabe von **mysql dump --no-data db_name tbl_name1 ....** Damit ist es möglich, die Datenbankstruktur auf einer neuen Maschine aufzubauen, und ohne Daten zu testen.
8. SELECT Statement im Original:

```
mysql> SHOW VARIABLES;
     mysql> EXPLAIN SELECT ...
            <output-from-EXPLAIN>
     mysql> FLUSH STATUS;
     mysql> SELECT ...
           <Hier könnte eine kurze Ausgabe eines MySQL Statements stehen>
     mysql> SHOW STATUS;
            <Ausgabe von SHOW STATUS>
```

9. Der Output von:

```
mysqladmin variables extend-status processlist
```

10. Fehlermeldungen von PHP, PERL, C, C++ oder anderen Werkzeugen
11. Die Privilegien der Tabellen (GRANT)
12. Vermutungen darüber, was der Fehler ist, zusammen mit kleinen Erläuterungen
13. Bei einem **parse error** bitte die Syntax genau überprüfen, erst dann den Fehler beschreiben.
14. Bei einer **corrupted table** Meldung den Speicherplatz der Festplatte, Partition und die Quotas überprüfen und mit in den BUG-Report schreiben.

Wie Sie sehen, ist viel Arbeit zu tun, bevor man auf Fragen auch qualifizierte und präzise Antworten durch den Support erhalten kann. In der Praxis stellt sich heraus, daß bei der Abarbeitung dieser Checkliste viele Probleme sich von alleine erledigen.

Besonders Entwickler - freundlich ist der folgende BUG Report, der mit dem phpMyAdmin Toolkit ganz einfach über die Funktion "DUMP" zu generieren ist:

```
I have a simple table, EventDateTbl that contains the following columns:

mysql> describe EventDateTbl;
+------------+---------+------+-----+------------+-------+
```

```
| Field      | Type    | Null | Key | Default    | Extra |
+------------+---------+------+-----+------------+-------+
| event_date | date    |      | PRI | 0000-00-00 |       |
| type       | int(11) |      | PRI | 0          |       |
| event_id | int(11)   |      | PRI | 0          |       |
+------------+---------+------+-----+------------+-------+
3 rows in set (0.00 sec)
```

A simple date range select seems okay:

```
mysql> select event_date,type,event_id from EventDateTbl
WHERE event_date >= "1999-07-01" AND event_date < "1999-07-15" ORDER BY
event_date;
+------------+--------+----------+
| event_date | type | event_id |
+------------+--------+----------+
| 1999-07-10 | 100100 | 24 |
| 1999-07-11 | 100100 | 25 |
| 1999-07-13 | 100600 | 0 |
| 1999-07-13 | 100600 | 4 |
| 1999-07-13 | 100600 | 26 |
| 1999-07-14 | 100600 | 10 |
+------------+--------+----------+
6 rows in set (0.00 sec)
```

And, if I add a single type to the WHERE, it also seems to work:

```
mysql> select event_date,type,event_id from EventDateTbl
WHERE event_date >= "1999-07-01" AND event_date < "1999-07-15"
AND (type=100600) ORDER BY event_date;
+------------+--------+----------+
| event_date | type   | event_id |
+------------+--------+----------+
| 1999-07-13 | 100600 | 0        |
| 1999-07-13 | 100600 | 4        |
| 1999-07-13 | 100600 | 26       |
| 1999-07-14 | 100600 | 10       |
+------------+--------+----------+
4 rows in set (0.01 sec)
```

But, if make it select on two different types, I get odd looking results with repeating results, and if I add another type=n in the query, I'll get three results, etc.:

```
mysql> select event_date,type,event_id from EventDateTbl
WHERE event_date >= "1999-07-01" AND event_date < "1999-07-15"
AND (type=100600 OR type=100100) ORDER BY event_date;
+------------+--------+----------+
| event_date | type   | event_id |
+------------+--------+----------+
| 1999-07-10 | 100100 | 24       | <-- July 10th
| 1999-07-11 | 100100 | 25       |
| 1999-07-13 | 100600 | 0        |
| 1999-07-13 | 100600 | 4        |
| 1999-07-13 | 100600 | 26       |
| 1999-07-14 | 100600 | 10       |
| 1999-07-10 | 100100 | 24       | <-- July 10th DUP, despite order by Date
| 1999-07-11 | 100100 | 25       |
```

```
| 1999-07-13 | 100600 | 0        |
| 1999-07-13 | 100600 | 4        |
| 1999-07-13 | 100600 | 26       |
| 1999-07-14 | 100600 | 10       |
+------------+--------+----------+
12 rows in set (0.01 sec)
```

Per another suggestion, I tried changing the 'type' clause to be:
      AND type IN (100600,100100)
but the results are the same.

This is odd to me because I have an ORDER BY that should put everything in date order, but that's not the case. I first noted this with a slightly more complex query (the one I really want to work, but I couldn't even get the trivial one to work).

```
mysql> select event_date,type,COUNT(type) from EventDateTbl
WHERE event_date >= "1999-07-01" AND event_date < "1999-07-15"
AND (type=100600 OR type=100100) GROUP BY event_date,type;
+------------+--------+-------------+
| event_date | type   | COUNT(type) |
+------------+--------+-------------+
| 1999-07-10 | 100100 | 1           |
| 1999-07-11 | 100100 | 1           |
| 1999-07-13 | 100600 | 3           |
| 1999-07-14 | 100600 | 1           |
| 1999-07-10 | 100100 | 1           |
| 1999-07-11 | 100100 | 1           |
| 1999-07-13 | 100600 | 3           |
| 1999-07-14 | 100600 | 1           |
+------------+--------+-------------+
8 rows in set (0.08 sec)
```

When I specify NO "type=nnn" in the WHERE CLAUSE, or when I have just ONE "AND type=nnn" it works fine, but if I put two types in, separated by ORs, I get the odd results.

What am I doing wrong?

Thanks,
David Wall
....
Here's the table dump for those that are interested:

```
# MySQL dump 6.0
#
# Host: localhost Database: eastside
#--------------------------------------------------------
# Server version 3.22.25-log



#
# Table structure for table 'EventDateTbl'
#
CREATE TABLE EventDateTbl (
  event_date date DEFAULT '0000-00-00' NOT NULL,
  type int(11) DEFAULT '0' NOT NULL,
  event_id int(11) DEFAULT '0' NOT NULL,
```

```
  PRIMARY KEY (event_date,type,event_id),
  KEY event_id (event_id)
);

#
# Dumping data for table 'EventDateTbl'
#
INSERT INTO EventDateTbl VALUES ('1999-07-10',100100,24);
INSERT INTO EventDateTbl VALUES ('1999-07-11',100100,25);
...
INSERT INTO EventDateTbl VALUES ('1999-09-19',100100,37);
INSERT INTO EventDateTbl VALUES ('2000-12-18',100700,38);
```

Kommerzieller Support für MySQL wird u.a. von den Programmierern von **MySQL** auf http://www.mysql.com angeboten. Hierbei wird zwischen verschiedenen Level unterschieden:

- Unverbindliche Betreuung über die Mailing-Listen. Viele Probleme werden dort diskutiert, die typisch für Einsteiger und Fortgeschrittete sind. Für verschiedene Themen sind entsprechende Listen angelegt, die man abonnieren sollte. Hier finden sich auch viele Betreuer von **MySQL** Datenbanken, die auch Fragen von Einsteigern beantworten. In vielen Fällen werden Fragen auch von den Entwicklern selber beantwortet.
- Verbindliche Betreuung via E-Mail. Dieser Service ist kostenpflichtig und sollte von Entwicklern, die mit **MySQL** kommerzielle Projekte entwickeln, unbedingt in Anspruch genommen werden. Eine Lizenz für **MySQL** ist für kommerzielle Projekte erforderlich. **MySQL** ist keine lizenzfreie Software, auch wenn Sie mit Quellcode geliefert wird.
- Login-Support. Hier können Entwickler auf die Erfahrungen der Entwickler selber zurückgreifen und sich die Software Online via Internet auf ihrem Server installieren lassen. Für Systeme, die mit einer ISDN-Karte und einer DIAL-ON-DEMAND Anbindung über einen beliebigen Provider angebunden sind, ist Online Administration ebenfalls möglich.
- Archive von Beiträgen. Diese Archive sollten Sie bei jeder Frage zuerst mit Hilfe der Suchmaschine durchforsten. Hier haben Sie schnellen Zugriff auf einen riesigen Fundus von Beiträgen. Siehe unter http://www.tryc.on.ca/archives/mysql/ oder auch http://www.bitmechanic.com/mail-archives/mysql/








15.3 Kommerzieller Support in Deutschland

Unverbindlicher Support in deutscher Sprache wird auf der Webseite http://www.rent-a-database.de angeboten. Auch hier wird zwischen verschiedenen Level unterschieden:

- Unverbindliche Betreuung über die Mailing-Listen. Viele Probleme werden dort diskutiert, die typisch für Einsteiger und Fortgeschrittete sind. Für verschiedene Themen sind entsprechende Listen angelegt, die man abonnieren sollte. Hier finden sich auch viele Betreuer von **MySQL** Datenbanken, die auch Fragen von Einsteigern beantworten. In vielen Fällen werden Fragen auch von Entwicklern verschiedener Projekte beantwortet.
- Verbindliche Betreuung über die Mailing Listen. Hier werden alle Fragen innerhalb eines bestimmten Zeitraumes (typischerweise ein paar Stunden bis zu einem Tag) garantiert beantwortet. Dieser Service ist kostenpflichtig. Die Antworten werden allerdings auch anderen Mitgliedern von Mailing-Listen zur Verfügung gestellt.
- Archive von Beiträgen. Diese Archive sollten Sie bei jeder Frage zuerst mit Hilfe der Suchmaschine durchforsten. Hier haben Sie schnellen Zugriff auf einen riesigen Fundus von Beiträgen. Siehe unter http://www.tryc.on.ca/archives/mysql/
- Online Installationssupport. Hier wird nach Wunsch Installationssupport für MySQL, ADABAS, PostgreSQL, Apache, PHP3, CGI_BIN´s für LINUX und SUN SOLARIS 2.5/2.6/2.7 geleistet. Für Server mit ISDN-Anbindung ist Fernwartung über ISDN oder auch Internet möglich.
- Exklusive Betreuung von Projekten. Hier wird exklusiv Support für MySQL, ADABAS, PHP3, PERL, LINUX und SOLARIS von Entwicklern geleistet.







15.4 MySQL Serverhosting

MySQL Serverhosting wird von vielen verschiedenen Anbietern in Deutschland angeboten. Hier muß man zwischen verschiedenen Arten der Installation unterscheiden. Hosting mit Datenbank Sharing und Hosting mit exklusiver Datenbank. Bei Datenbank Sharing erhalten Sie Zugriff auf für Sie reservierten Tabellen in einer größeren Datenbank, auf welche aber nur Sie Zugriff haben. Für fast alle kleineren Projekte ist dieses Angebot ausreichend. Bei Anbietern von Hosting mit exklusivem Datenbank-Zugriff erhalten Sie auf einem Server eine eigenen **mysqld** installiert, auf welchen Sie exklusiv Zugriff haben. Dort können Sie auch sämtliche Zugriffe auf Tabellen selber verwalten.

Anbieter mit **MySQL** Sharing:

- http://www.Rent-A-Database.de

Anbieter mit exklusivem Datenbankzugriff auf eine eigene **MySQL-Datenbank**:

- http://www.rent-a-database.de







15.5 Schulungen MySQL / PHP 4.0

Schulungen in Deutschland werden von folgenden Firmen angeboten:

- http://www.rent-a-database.de







15.6 Fragen und Antworten - Mailinglisten

In Ergänzung zu diesem Handbuch ist es für Entwickler sinnvoll, sich auf einer der Mailing-Listen einzutragen. Gegenseitige Hilfe und Austausch von wichtigen Informationen stehen im Vordergrund. Zu **MySQL** sind verschiedene Mailinglisten verfügbar, welche in der Originaldokumentation aufgelistet und beschrieben sind und von dem Entwicklerteam verwaltet werden. Es steht auch eine deutsche Mailingliste zur Verfügung. Erreichbar ist diese unter:

```
mysql-de-request@lists.4t2.com (Listmanager)
bzw.
mysql-de@lists.4t2.com
oder
mysql-de@rent-a-database.de
```

Um sich in die Mailingliste einzutragen, muß man eine E-Mail an:

```
mysql-de-request@lists.4t2.com
oder
mysql-de-request@rent-a-database.de
```

senden, die als **Inhalt** (nicht in das Subject)

```
subscribe eMail - Adresse
```

im Body stehen hat. Bug-Reports sind an die Adresse

```
mysql@lists.mysql.com mit dem Subject: "documentation suggestion: Thema"
```

zu senden. Bevor man sich aber an die Entwickler selber wendet, sollte man stets einmal in den Mailing Liste, oder im Internet auf der Site http://www.deja.com nach dem Problem suchen. Hierbei sollten möglichst bezeichnende englischsprachige Begriffe als Sucheinträge verwendet werden. In den meisten Fällen dürfte sich dann bereits ein Hinweis auf einen möglichen Fehler ergeben. Probleme, die auf dem Hauptserver http://www.tcx.se als **todo** oder als **known bugs**

Es gibt eine ganze Reihe von Listen, die spezifische Themen bezüglich MySQL behandeln:

Um sich auf der Haupt **MySQL** Liste einzutragen, schicken Sie bitte eine Nachricht an mit Ihrer E-Mail-Adresse im Nachrichtentext (nicht im Subject) an mysql-subscribe@lists.mysql.com

Um sich auf der Haupt **MySQL** Liste **auszutragen**, schicken Sie bitte eine Nachricht an mit Ihrer E-Mail-Adresse im Nachrichtentext (nicht im Subject) an mysql-unsubscribe@lists.mysql.com

Es existieren folgende MySQL Mailing Listen, auf denen man sich analog zum obigen Beispiel Ein-und Austragen kann.

- **announce** Diese Liste ist für Neuankündigungen von **MySQL** Versionen und ähnlichen Programmen. **MySQL** User sollten sich auf dieser Liste eintragen.
- **mysql** Dieses ist die wichtigste Diskussionsliste. Einige Themen sind allerdings besser in den spezialisierten Listen aufgehoben.
- **mysql-digest** Dieses ist die tägliche Zusammenfassung der **mysql** - Liste .
- **java** Hauptthema dieser Liste sind JAVA und JDBC-Treiber
- **java-digest** Dies ist die tägliche Zusammenfassung der **java** - Liste.
- **win32** Hier werden alle Themen bezüglich **MySQL** und Microsoft Betriebssystemen diskutiert.

- **win32-digest** Dies ist die tägliche Zusammenfassung der **java** - Liste.
- **myodbc** Hauptthema dieser Liste sind ODBC-Treiber
- **myodbc-digest** Dies ist die tägliche Zusammenfassung der **myodbc** - Liste.
- **msql-mysql-modules** Auf dieser Liste wird Perl Support für **MySQL** diskutiert.
- **msql-mysql-modules-digest** Dies ist die tägliche Zusammenfassung der **msql-mysql-modules** - Liste.
- **developer** Hier finden sich alle Entwickler von **MySQL** - Code zusammen.
- **developer-digest** Dies ist die tägliche Zusammenfassung der **developer** - Liste.

Es reicht gewöhnlich aus, wenn man sich täglich einmal alle Beiträge gesammelt zusenden läßt...






16. Das Jahr 2000 Problem

MySQL ist Jahr 2000 fest ! **MySQL** verwendet das **UNIX**-Format bei der Verarbeitung der Zeit- und Datumsfunktionen. Es gibt stets die Zahl der Sekunden seit dem 1.1.1970 an. Das Jahr 2000 Problem ist bei keinem UNIX Derivat ein Problem. Interpretationsbedürftig ist eventuell nur die Ausgabe der RTC (Real Time Clock) des BIOS bei INTEL Hardware. So findet inzwischen bei fast allen UNIX Derivaten eine Korrektur bei alten BIOSen statt. Wenn die Jahresangabe kleiner als 70 ist, das ist bei zweistelligen Jahresangaben ab dem 1.1.2000 so, dann werden 2000 hinzuaddiert, wenn die Jahresangabe größer oder gleich 70 ist, dann wird nur 1900 hinzuaddiert.

UNIX-Programme besitzen nur dann ein Jahr 2000 Problem, wenn keine vierstelligen Datumsangaben in dem Anwendungsprogramm selber möglich sind.

MySQL kann eventuelle Probleme mit Anwendungsprogrammen aber noch auf andere Weise eventuell lösen. Hierzu hat **MySQL** Datenformat eingeführt, welches das Jahr stets nur zweistellig anzeigt, jedoch intern als vierstelliges Format abspeichert und mit diesem Format auch rechnet. Falls also die Anwendungssoftware nur 2 - stellig arbeitet, wird **MySQL** trotzdem korrekt rechnen und die korrekten Resultate an die Anwendungssoftware übergeben.

Folgende Ausgabe zeigt, daß MySQL bis zum Jahr 2030 keine Probleme hat:

```
mysql> DROP TABLE IF EXISTS y2k;
mysql> CREATE TABLE y2k (date date, date_time datetime, time_stamp
timestamp);
mysql> INSERT INTO y2k VALUES ("1998-12-31","1998-12-31
23:59:59",19981231235959);
mysql> INSERT INTO y2k VALUES ("1999-01-01","1999-01-01
00:00:00",19990101000000);
mysql> INSERT INTO y2k VALUES ("1999-09-09","1999-09-09
23:59:59",19990909235959);
mysql> INSERT INTO y2k VALUES ("2000-01-01","2000-01-01
00:00:00",20000101000000);
mysql> INSERT INTO y2k VALUES ("2000-02-28","2000-02-28
00:00:00",20000228000000);
mysql> INSERT INTO y2k VALUES ("2000-02-29","2000-02-29
00:00:00",20000229000000);
mysql> INSERT INTO y2k VALUES ("2000-03-01","2000-03-01
00:00:00",20000301000000);
mysql> INSERT INTO y2k VALUES ("2000-12-31","2000-12-31
23:59:59",20001231235959);
mysql> INSERT INTO y2k VALUES ("2001-01-01","2001-01-01
00:00:00",20010101000000);
mysql> INSERT INTO y2k VALUES ("2004-12-31","2004-12-31
23:59:59",20041231235959);
mysql> INSERT INTO y2k VALUES ("2005-01-01","2005-01-01
00:00:00",20050101000000);
```

```
mysql> INSERT INTO y2k VALUES ("2030-01-01","2030-01-01
00:00:00",20300101000000);
mysql> INSERT INTO y2k VALUES ("2050-01-01","2050-01-01
00:00:00",20500101000000);
mysql> SELECT * FROM y2k;
+------------+---------------------+----------------+
| date       | date_time           | time_stamp     |
+------------+---------------------+----------------+
| 1998-12-31 | 1998-12-31 23:59:59 | 19981231235959 |
| 1999-01-01 | 1999-01-01 00:00:00 | 19990101000000 |
| 1999-09-09 | 1999-09-09 23:59:59 | 19990909235959 |
| 2000-01-01 | 2000-01-01 00:00:00 | 20000101000000 |
| 2000-02-28 | 2000-02-28 00:00:00 | 20000228000000 |
| 2000-02-29 | 2000-02-29 00:00:00 | 20000229000000 |
| 2000-03-01 | 2000-03-01 00:00:00 | 20000301000000 |
| 2000-12-31 | 2000-12-31 23:59:59 | 20001231235959 |
| 2001-01-01 | 2001-01-01 00:00:00 | 20010101000000 |
| 2004-12-31 | 2004-12-31 23:59:59 | 20041231235959 |
| 2005-01-01 | 2005-01-01 00:00:00 | 20050101000000 |
| 2030-01-01 | 2030-01-01 00:00:00 | 20300101000000 |
| 2050-01-01 | 2050-01-01 00:00:00 | 00000000000000 |
+------------+---------------------+----------------+

13 rows in set (0.00 sec)
mysql> DROP TABLE y2k;
```

Das bedeutet, daß MySQL ab dem Jahr 2030 erst wieder Probleme hat. Der Grund liegt in einem Wrap-Around des (internen) 31 Bit Wertes für die Zahl der Sekunden seit dem 1.1.1970. Leider haben viele Datenbankhersteller diesen Test noch nicht durchgeführt. Falls also bei einer Versicherung, Bank o.ä. die Verfallsdaten von langfristigen Verträgen gespeichert werden müssen, gibt es erhebliche Probleme (hehe !), den Normalanwender sollte dies aber nicht interessieren.






17.1 Foreign Keys

Foreign Keys werden in SQL im Allgemeinen nicht verwendet, um Tabellen zu verknüpfen (JOIN), sondern um sicherzustellen, daß die Integrität der verschiedenen einiger Spalten von Tabellen sichergestellt ist. Beispielsweise kann man hiermit erzwingen, daß wenn ein Datensatz verändert wird, in einer anderen Tabelle entsprechende Veränderungen automatisch vorgenommen werden. Es gibt aber auch viele Gründe, diese möglichst wenig einzusetzen:

- Foreign Keys verkomplizieren das allgemeine Datenbankhandling. Die Einfachheit von **MySQL**, z.B. im **Intranet** eine Datenbank zu pflegen (SHOP), und täglich in das Internet zu kopieren, wäre dahin.
- Die Geschwindigkeit von INSERT und UPDATE Stements wäre ebenfalls gefährdet, da die Datenbank nach einem INSERT oder UPDATE alle foreign keys einzeln duchlaufen muß. Im Allgemeinen werden ohnehin INSERTS stets in den richtigen Tabellen (mehrere INSERT/UPDATES in mehreren Tabellen gleichzeitig) durchgeführt. Also keine Panik, man kann diese Konstrukte durch **Workarounds** in den Griff bekommen. Abfragen und Verknüpfungen über mehrere Tabellen sind aber stets erlaubt, damit keine Mißverständnisse aufkommen.
- Backups und Restores werden fast unmöglich gemacht. Der einfache Vorgang, eine einzelne Tabelle bei Mangel an Datenintegrität (Festplattenfehler) einfach zu löschen, und dann von der Backup-Platte oder dem Tape zu restaurieren, funktioniert bei Datenbanken mit foreign keys nicht mehr. Es muß in diesen Fällen dann immer die gesamte Datenbank restauriert werden, und es muß eine bestimmte Reihenfolge peinlich genau eingehalten werden. Bei großen Datenbanken, die über mehrere Millionen Einträge verfügen, ist dann mit längeren Ausfällen zu rechnen. Beim Einsatz im Internet muß die Datenbank jedoch den **"mission critical"** Anforderungen entsprechen.
- Bei foreign keys passiert es recht häufig, daß zirkuläre (rekursive) Bezüge entstehen, die zwar erlaubt sind, jedoch verhindern, daß man eine Tabelle so einfach mit einem CREATE Befehl erzeugen kann. Hierfür sind oft sehr komplexe Statements notwendig.
- Und nun einge gute Nachricht: **MySQL** wird in absehbarer Zeit um foreign keys erweitert, im SQL Parser sind diese schon enthalten.







17.2 ON DELETE, ON UPDATE

Diese Kommandos gehören in die Kategorie Trigger und führen Statements in Abhängigkeit von Änderungen im Datenbestand aus. MySQL unterstützt keine Trigger, da deren Einsatz oft nur in Verbindung mit **stored procedures** sinnvoll ist. Durch externe PERL oder PHP-Scripte ist es möglich, erheblich mächtigere Funktionen zu implementieren, als es allein mit **triggern** oder **stored procedures** möglich wäre.







17.3 UNION

UNION ermöglicht bei Subselects die Verknüpfung von Mengen miteinander. Da MySQL und ACCESS keine Subselects unterstützen, sind auch UNION´s nicht notwendig:

In ANSI SQL 92 werden Mengen so verknüpft:

```
SELECT db1.tabelle .....
UNION
SELECT db2.tabelle ....
ORDER BY name
```

Das funktioniert so bei MySQL nicht. Im Grunde wird bei dem Statement **UNION** in anderen Datenbanken durch den Pre-Prozessor ebenfalls eine temporäre Tabelle angelegt, nur, daß der User davon nichts merkt.

Bei MySQL muß stattdessen eine schnelle, temporäre Tabelle angelegt werden, die bei den neuen MySQL Versionen im RAM angelegt, und evtl. auch automatisch auf den Swap-Speicher ausgelagert wird. Hierzu sehen Sie bitte im Kapitel CREATE und dort bei **table_options** nach. Darin ist eine Option **HEAP** erwähnt, die superschnelle Tabellen im RAM anlegt. Das Statement SET, in Kapitel SET ermöglicht Tuning während der Laufzeit.

```
CREATE TABLE tmp1 (tabelle ....);
INSERT INTO tmp1 SELECT db1.tabelle .....;
INSERT INTO tmp1 SELECT db2.tabelle .... ;
SELECT * FROM tmp1 WHERE ... ORDER BY name;
```

Danach kann man entweder die ganze Tabelle Löschen, oder nur deren Inhalte, damit man diese nochmal verwenden kann.






17.4 Locks

Locks bei Schreibzugriffen auf ganze Tabellen bremsen SQL Datenbanken ungeheuer aus. Daher hat man in guten Datenbanken zusätzlich sowohl Row-Level als auch Column-Level Locks zur Verfügung. Microsoft SQL 6.5 kennt ROW-LEvel Locks übrigens auch nicht.

Wenn zudem wegen den Foreign Keys noch Locks in weiteren Tabellen angelegt werden müssen, dann brechen fast alle Datenbanken völlig ein.

Besonders bemerkbar mach sich dies, wenn zudem noch serverseitige Cursor eingesetzt werden, und jemand zuerst scrollt, bevor er einen Datensatz ändert. In diesem Moment wird ohne Caching im Treiber (ODBC/JDBC Klasse 4) ständig ein Lock auf eine oder mehrere Tabellen gesetzt. Beherrscht die Datenbank noch kein ROW-Level Locking, so muß der Lock auf die Tabelle, bzw. beim Einsatz von Foreign Keys auf mehrere Tabellen gesetzt werden.

Andere User können währenddessen nichts eingeben oder ändern. Problematisch ist aber auch der Fall, wenn die Datenbank ROW-Level oder Column-Level Locks beherrscht, aber die Locks falsch gesetzt werden. In diesem Fall gibt es zuviele Locks in der Datenbank, was die Performance stark herunterdrückt. Ohne ein vernünftiges Transaktionsmanagement, welches Schreibzugriffe verzögert, um sie dann gesammelt in die Datenbank zu schreiben, bei vielen gleichzeitigen INSERT/UPDATES die Datenbank dann stillstehen.

Die Entwickler von **MySQL** sind daher einen anderen Weg gegangen. Unter UNIX und NT (nicht Windows 95/98) wird jedem Zugriff auf die Datenbank ein Thread zugeordnet (eine Art Unterprogramm).

Es gibt Threads, die Lesen, und Threads, die in die Datenbank scheiben. Wenn nun viele User gleichzeitig die Datenbank auslesen möchten, können sie dies ohne Probleme tun. Wenn nun einige User in die Datenbank schreiben möchten, so werden für jeden gewünschten Schreibzugriff ein exklusiver Thread zugeordnet. Diese Threads erhalten, bis der Schreibzugriff auf die Datenbank abgearbeitet ist, einen exklusiven Lock auf die ganze Tabelle, während alle anderen Schreib-Threads warten müssen. Normalerweise würde dies den Stillstand für alle anderen User bedeuten, bei anderen Datenbanken zumindest.

MySQL besitzt aber ein ausgefeiltes Thread-Handling. Es stellt alle anderen Schreib-Threads solange in einer Art Task-Manger (oder Scheduler, wem dieser Begriff mehr sagt) zurück, bis kein andere Lese-oder Schreib Thread mehr auf die Datenbank zugreift. Wenn also die Datenbank stark gefordert ist (100 Abfrage/Sekunde), dann könnte es passieren, daß ständig Threads lesend zugreifen, und die Schreib-Threads nicht ausgeführt werden.

Darum haben die Entwickler in SQL das Statement **LOW_PRORITY** bzw. **DELAYED** bei SELECT, INSERT, UPDATE.. (DELAYED) ..Statements (alle DML Befehle) hinzugefügt, damit einige Treads gegenüber anderen priorisiert, also bevorzugt abgearbeitet werden können. Man kann somit entweder den Lesevorgängen oder den Schreibvorgängen eine hohe Priorität geben. Im Allgemeinen gibt man den Schreibvorgängen eine hohe Priorität vor allen Lesevorgängen, zumal die Schreibvorgänge auch kaum Zeit in Anspruch nehmen, im Vergleich zu den häufiger länger dauernden Abfragen. Wer z.B. INSERT DELAYED benutzt, um Datensätze einzufügen, der wird unter **MySQL** nicht blockiert, auch wenn die Datenbank unter hoher Last steht. Der Grund ist folgender: Für jeden Schreibzugriff wird ein eigener Thread angelegt, der den INSERT Befehl entgegen nimmt und dem Benutzer OK zurückmeldet. Der Thread wird dann ausgeführt, wenn die Datenbank kurzzeitig ohne Last ist. Das Verfahren ist also ähnlich den von anderen SQL-Datenbanken bekannten Transaktionen, wird nur halt auf Thread-Ebene und auch nur im RAM ausgeführt. Bei einem Absturz sind dann die Änderungen in den noch nicht abgearbeiteten Threads (meist sehr wenige) also verloren.

In der Praxis ist **MySQL** auch ohne ROW/COLUMN-Locking um den Faktor 5-10 schneller, als z.B. ORACLE, bezogen auf den gesamten Durchsatz. Nach den Erfahrungen beim Aufbau von großen Datenbanken im Internet (Ein Kunde hat 750.000 CD-ROM's zum Verkauf (Siehe http://www.cd-special.de) sind die Antwortzeiten bei MySQL trotz PERL-Frontend sehr kurz. Ein Test mit ORACLE 8i hat im Stresstest ca. Faktor 5-10 längere

Antwortzeiten erbracht.

Ein einfacher Pentium mit 128 MB RAM hätte da nicht mehr ausgereicht, sondern es hätte ein DEC-ALPHA Cluster mit LOAD-Balancing zum Einsatz kommen müssen. **MySQL** kann aber auch Locks auf Tabellen-Ebene ausführen. Siehe hierzu das Kapitel Lock/Unlock von Tabellen. Man kann aber auch ohne Veränderungen des SQL-Codes die Datenbank dazu veranlassen, die Prioritäten anders zu setzen: Hier ein entsprechender Ausschnitt aus der Support-Datenbank der Entwickler von Monty selber geschrieben:

```
Why not use:

mysqldump --flush-logs --lock-tables

set SQL_LOW_PRIORITY_UPDATES=1;

You can instead start mysqld with:

--low-priority-updates

or use:

[INSERT | UPDATE | DELETE] LOW_PRIORITY ...

The problem with locks are the following:

If you do (in this time order)

#1 LOCK TABLES test READ
#2 LOCK TABLES test WRITE
#3 LOCK TABLES test READ


#1 has a read lock on the table.
#2 has a 'wait for write lock' on the table
#3 has a 'wait for read lock' on the table.

#3 has to wait for #2 as write locks has (normally) higher privilege
than read locks.

The reason for this is if you would allow SELECT's to proceed on
tables where there is a 'waiting write lock', then if you issue new
selects the whole time, it's likely that there will never be a time when
the table is 'free' and the clients that wants to update will starve
to death.

In the case of INSERT's, you can probably avoid this by using 'INSERT
DELAYED ...'

Regards,
Monty
```






17.5 Cursors

Cursor sind eine komplexe Angelegenheit. Es gibt Serverside, Clientside, Forward-Only .... Cursor. Danach ist es eine Frage des Anwendungs-Interfaces, also den Fähigkeiten des ODBC-Treibers oder JDBC Treibers. MyODBC unterstützt in Verbindung mit Microsoft Applikationen **clientside cursors**, auch als **front-end cursors** beschrieben. Diese funktionieren mit allen Microsoft Anwendungen und Programmiersprachen. Mi ACCESS z.B. als Frontend können sie ganz einfach durch Datensätze in der MySQL Datenbank durchscrollen. Dasselbe betrifft auch alle Frontends, die mit Microsoft Programmen entwickelt wurden.

Serverside Cursor werden von MySQL nicht unterstützt, da diese oft erhebliche Performanceprobleme auf dem Server bereiten. Bei vielen simultanen Clients, wie sie oft im Internet auftreten können, sind Serverside Cursor das Ende jeder SQL Datenbank. Der Grund dafür liegt einfach darin, daß alle Statements über einen Postprozessor der Datenbank laufen, der dann die Cursor mit der Ausgabe der Statements verknüpft, und daraufhin entscheidet, welche Daten dem Client übermittelt werden. Das kostet Auslagerungsspeicher und viel CPU-Zeit. Clientside Cursor schonen erheblich Serverresourcen, da diese neuerdings auch Caching - Mechanismen unterstützen. Bei Internet-Anwendungen sind also in jedem Falle eine hochkarätige Datenbank, wie z.B. ORACLE, SYSBASE oder MS-SQL mit **serverside cursors** völlig ungeeignet. Unter JDBC gibt es ebenfalls **clientside cursor** für MySQL, die auch völlig problemlos funktionieren. Einige Hersteller bieten für MySQL erweiterte Bibliotheken an, die erhebliche Erweiterungen und Vorteile gegenüber den RMI JDBC Bibliotheken von SUN bieten. Siehe hierzu auch Kapitel Cursor unter MySQL. Wenn Sie sich für eine Art Cursor unter PHP3 interessieren, mit denen man seitenweise Daten aus MySQL anzeigen und ändern kann, sollten Sie sich im Kapitel PHP3 Tutorial umschauen.







17.6 Views

Views sind nicht weiter, als temporäre Tabellen. In der Sprachreferenz MySQL wird auf die Syntax von temporären Tabellen genau eingegangen. Die Statements werden dadurch zwar etwas komplexer, funktionell können diese aber VIEWS vollständig ersetzen.







17.7 Subselects

Nun, um es vorweg zu nehmen, ACCESS braucht sie auch nicht (?). Subselects kann man durch JOINS recht einfach umschreiben. Es erhöht zwar nicht gerade die Lesbarkeit, ein (self) JOIN ist aber ein vollständiger Ersatz. In der nächsten Version ist aber damit zu rechnen, daß Subselects untestützt werden. Übrigens gibt es in vielen Datenbanken Beschränkungen der Schachteltiefe auf nur wenige **SELECT * FROM WHERE ...SELECT * FROM ... WHERE** Statements. Es sind auch oft nicht alle Kombinationen mit anderen Parametern erlaubt. Daher sind Subselects nicht unbedingt das K.O. Kriterium für die Auswahl von (nicht MySQL) Datenbanken, inbesondere nicht für User, die von ACCESS her sich **MySQL** widmen möchten, da hier einige Statements nicht zugelassen sind.

Ein Beispiel ist z.B. der Ausdruck:

```
SQL> SELECT NAME FROM KURS_1 UNION SELECT NAME FROM KURS_2
```

Hiermit wird die Vereinigungsmenge der Kursteilnehmer aus beiden Kursen ausgegeben. **MySQL** kan dieses mit SELF-JOINS umschreiben.

Ein weiteres Beispiel ist die Schnittmenge der Teilnehmer aus beiden Kursen:

```
SQL> SELECT NAME FROM KURS_1 INTERSECT SELECT NAME FROM KURS_2
```

Ein anderes Beispiel ist die Menge aller Kursteilnehmer, die nicht in dem zweiten Kursus sind:

```
SQL> SELECT NAME FROM KURS_1 MINUS SELECT NAME FROM KURS_2
```

Diese Schreibweise ist zwar sehr elegant, aber auch einfach mit SELF-JOINS zu umschreiben. Hier die Schreibweise in anderen SQL Dialekten:

```
SELECT * FROM table1 WHERE id IN (SELECT id FROM table2);
SELECT * FROM table1 WHERE id NOT IN (SELECT id FROM table2);
```

Hier der Ersatz in MySQL

```
SELECT table1.* FROM table1,table2 WHERE table1.id=table2.id;
SELECT table1.* FROM table1 LEFT JOIN table2 ON table1.id=table2.id where table2.id
IS NULL
```

Komplexere Statements sollte man mit temporären Dateien umschreiben. Folgende Statements funktionieren jedoch.

```
INSERT ... SELECT ... and REPLACE ... SELECT ...
```

Wie man sieht, gibt es kaum gute Gründe, **ausgewachsene** SQL Datenbanken einzusetzen. Ein Grund, MySQL **nicht** zu Beginn einzusetzen, ist dies aber nicht. Die SELECT Statements von MySQL sind ohne Probleme auf andere SQL Datenbanken portierbar. Die Aufwätskompatibilität ist also gewährleistet.







17.8 Joins

Es gibt bei vielen Datenbanken einige JOIN - Formen nicht. Auch Microsoft SQL 7.0 besitzt hier z.B. erhebliche Abweichungen vom ANSI SQL 92 Standard. MySQL besitzt alle Möglichkeiten der Verknüpfung von Tabellen, jedoch könnte es sein, daß ein Statement dann nicht so elegant formuliert werden kann, wie bei anderen Datenbanken. Da in den Mailing-Listen immer wieder dieselben Schwierigkeiten auftreten, ist das Kapitel JOINS gegenüber dem original MySQL Handbuch erheblich erweitert worden, sodaß sie ein kleines Tutorial darstellen.







17.9 Transaktionen

Transaktionen fassen viele Lese und Schreibzugriffe zu einem Prozeß zusammen, der dann sehr schnell ausgeführt wird. Hierzu ist ein Transaktionspuffer notwendig, der verzögert alle Änderungen in die Datenbank schreibt. Mit Transaktionen kann man das Schreiben von vielen Clients in eine Datenbank drastisch beschleunigen. Zumindest hat dann der User den Eindruck, die Datenbank würde schnell arbeiten. In Wirklichkeit werden diese INSERT und UPDATE Befehle in eine Art QUEUE geschoben, die dann langsam abgearbeitet wird. Hierzu wird im Allgemeinen mit BeginTrans/CommitTrans und dem Verwerfen einer Transaktion RollbackTrans gearbeitet. CommitTrans sorgt dafür, daß die Änderungen auch tatsächlich durchgeführt werden. Leider können nur wenige Datenbanken Transaktionen dann noch durchführen, wenn verteilte Datenbanken, Tabellen in anderen Datenformaten (ACCESS), oder gleichzeitig Datensynchronisation eingesetzt werden müssen. Hier sollte man sich genauestens beim Hersteller informieren.

MySQL kann jedoch die Befehle, wie INSERT mit einer niedrigen Priorität versehen (**LOW_PRIORITY**), sodaß der Befehl erst dann ausgeführt wird, wenn **MySQL** eine kurze Pause hat. Dank der Multi-Thread Architektur ist das ohne Probleme möglich.

Wer allerdings mit ACCESS als Frontend, ODBC und **MySQL** arbeitet, der kann den Transaktionsmechanismus (Siehe obige Befehle aus ACCESS) von ACCESS selber nutzen.

Ein guter Workaround ist der Einsatz von **MySQL** zusammen mit PERL oder besser PHP3. Unter PHP3 kann man die Suchergebnisse in eine temporäre Datei schreiben, und in diesen dann mit Hilfe des PHP3 Interpreters blättern. Beispiele finden Sie im Kapitel Programmierung mit PHP3 3.0/4.0. Schreibvorgänge kann man logischerweise ebenso mit PHP3 3.0/4.0 speichern, und später unter Verwendung von des Ausdrucks **low_priority**, siehe auch LOW_PRIORITY, verzögert in die Datenbank schreiben. Im Grunde funktioniert der Transaktionsmechanismus anderer Datenbanken ebenfalls mit Hilfe eines vorgeschalteten Programmes, welches diese Transaktionen bündelt und optimiert. Angesichts der wenigen Zeilen zusätzlich, die man in PHP3 3.0/4.0 benötigt, um diesen Mechanismus zu implementieren, ist dies keine Hindernis, sondern eher ein Vorteil bei verteilten Datenbanksystemen, da man die temporären Dateien via FTP oder sogar via E-Mail über das Internet verbreiten und somit die Datenbanken synchronisieren kann. Auf diese Art und Weise werden sogar Standleitungen entbehrlich. Siehe hierzu auch das Kapitel Verteilte Datenbanken







17.10 Stored Procedures

Stored Procedures sind Makro´s oder auch Funktionen, bei denen Parameter übergeben werden können. Normalerweise werden diese bei SQL-Datenbanken mit **DECLARE** eingeleitet.

Sie sind sehr praktisch, weil man hiermit auch Personen, die kein SQL beherrschen, ermöglicht, z.B. regelmäßige Statistiken zu erstellen. Der Aufruf einer Zeile im SQL-Client reicht da schon aus. Stored Procedures in Verbindung mit z.B. Nested Cursors sorgen allerdings dafür, daß die Prozeduren unglaublich komplex werden. Hier ein kleiner Ausschnitt einer solchen mit **nested cursors** unter MS SQL 6.5 :

```
Create Procedure Maintain_Employees
As
/* First declare variables that are going to
be required in this procedure */
Declare     @dtPubDate   datetime,
            @nEmplCount  smallint,
            @nEmplID     empid,
            @nFirstHalf  smallint,
            @nRowCount   integer,
            @nSecondHalf integer,
            @nSQLError   integer,
            @nYtdSales   integer,
            @sLName      varchar(30),
            @sPubID      char(4),
            @sLastType   char(12),
            @sType       char(12)
/* Now declare the cursors to be used.
Note that because variables are used in the
where clause on the second cursor, it is not
required that the second cursor be declared inside the first.
Take advantage of this functionality so that unnecessary
declaring of cursors does not take place (this will
save resources on the server). */
Declare Cur_Empl Cursor
For   Select EMP_ID,    LNAME,
             PUB_ID
      From   EMPLOYEE
      Order By EMP_ID
Declare Cur_Titles Cursor
For   Select  TYPE,   PUBDATE, YTD_SALES
      From    TITLES
      Where   PUB_ID = @sPubID
Order By TYPE
/* Open the outer cursor and fetch the first row */
Open  Cur_Empl
Fetch Cur_Empl
Into  @nEmplID,   @sLName,
      @sPubID
/* Initialize counters */
Select      @nEmplCount = 0
While @@Fetch_Status = 0           /* Fetch only while there are rows left
*/
```

```
Begin
      /* increment counter */
      Select @nEmplCount = @nEmplCount + 1
      /* Return a result set to the front-end so that it knows
      what is happening */
      Select      @nEmplID,   @sLName
      If @sLName < 'D'  /* Skip all the D's by using a GOTO */
            Goto Fetch_Next_Empl
      /* Now open inner cursor and count the different types
      of books for this employee's publisher */
      Open Titles
      Fetch Titles
      Into  @sType, @dtPubDate, @nYtdSales
      /* Reset totals */
      Select @nFirstHalf = 0,
            @nSecondHalf = 0,
            @sLastType = NULL
      While @@Fetch_Status = 0
      Begin
            If @sType != @sLastType AND @sLastType != NULL
            Begin
                  /* Send back a total record to the front-end */
                  Select @sLastType, @nFirstHalf, @nSecondHalf
                  /* Reset totals */
                  Select @nFirstHalf = 0,
                        @nSecondHalf = 0
            End
            If @dtPubDate <= 30/95'
                  Select @nFirstHalf = @nFirstHalf + @nYtdSales,
                        @sLastType = @sType
            Else
                  Select @nSecondHalf = @nSecondHalf + @nYtdSales,
                        @sLastType = @sType
            Fetch Titles
            Into  @sType, @dtPubDate, @nYtdSales
      End
      Fetch_Next_Empl:        /* Label to skip inner loop */
      Fetch Cur_Empl
      Into  @nEmplID,   @sLName,
            @sPubID
End
/* Deallocate and close the cursors. Note that for a stored
procedure, this is really unnecessary because the cursor
will no longer exist once the procedure finishes execution.
However, it is good practice to leave the procedure cleaned up. */
Close Cur_Empl
Deallocate Cur_Empl
Deallocate Cur_Titles
/* Send total count of employees to front-end */
Select @nEmplCount
/* End proc */
Return 0
```

Wer nun immer noch nicht von diesen **stored procedures** in Verbindung mit **cursors** geheilt ist, der muß dann von einer einfachen Lösung unter MySQL, die es ebenso tut, auf die komplexeren unter ORACLE, MS-SQL 6.5/7.0 oder SYBASE umsteigen, und viel Geld in KNOW-HOW investieren. Für Internet-Datenbanken jedenfalls sind diese Prozeduren nicht notwendig.

Mit MySQL kann man ebenfalls mehrere SQL-Befehle zusammenfassen, und diese auf der Festplatte ablegen. Mit

Hilfe des (siehe Kapitel Batch-Modus) kann man diese Batch-Scripte durch den SQL-Client laden und ausführen lassen. Der wesentliche Unterschied ist, daß man leider noch keine Parameter übergeben kann. Da man die Batch-Scripte auch mit Hilfe der SHELL aufrufen kann, ist es aber so möglich, die Parameter an ein Shellscript übergibt, welches dann die SQL-Befehle zusammenfasst und dann von der SQL-Datenbank ausführen läßt. Ein einfacher, zuverlässiger und vor allem universeller Workaround. Zudem kann man diese Scripte auch über den CRON-Dämon zeitgesteuert starten lassen. Auch wenn **MySQL** offiziell keine **stored procedures** beherrscht, es gibt aber dank der freien Verfügbarkeit des Quellcodes die Lösung, die Sprache MySQL nach belieben zu erweitern ! Im Grunde sind **stored procedures** nur erfunden worden, da es damals noch nicht den Gedanken der OpenSource Bewegung gegeben hat. Mit dem Quellcode in Händen können Sie relativ einfach mit C-Kenntnissen MySQL erweitern. Diese neuen Prozeduren bzw. Funktionen laufen mit einer Geschwindigkeit ab, von der andere Datenbankhersteller nur träumen können.

Hier ein Ausschnitt aus dem originalen Handbuch:

```
15.1 Procedure analyse
analyse([max elements,[max memory]])

This procedure is defined in the Sql/sql_analyse.cc'. This examines the
result from your query and returns an analysis of the results.

max elements (default 256) is the maximum number of distinct values analyse
will notice per column. This is used by analyse to check if the optimal
column type should be of type ENUM.
max memory (default 8192) is the maximum memory analyse should allocate per
column while trying to find all distinct values. <p>
SELECT ... FROM ... WHERE ... PROCEDURE ANALYSE([max elements,[max memory]])

15.2 Writing a procedure.
For the moment, the only documentation for this is the source. :(

You can find all information about procedures by examining the following
files:

Sql/sql_analyse.cc'
Sql/procedure.h'
Sql/procedure.cc'
Sql/sql_select.cc'
```

Schreiben Sie sich doch einfach einen neuen SQL-Befehl. Da **MySQL** im Gegensatz zu allen anderen Datenbanken im Quellcode geliefert wird, können Sie sich Ihre eigenen Befehle implementieren, und die Datenbank noch einmal neu kompilieren. In vielen Datenbanken ist in dem Kapitel von **stored procedures** von **Compiler** die Rede. Damit keine Verwechselungen aufkommen... Das sind in Wirklichkeit keine Compiler, die die **stored procedures** in die Maschinensprache der CPU übersetzen, sondern in irgendeine andere interne Sprache der SQL-Datenbank, die dann aber nur interpretiert wird. Der Geschwindigkeitsunterschied zwischen MySQL C++ Prozeduren und Prozeduren bei anderen SQL-DAtenbanken dürfte zwischen 5-20 liegen. MySQL ist kein Spielzeug, sondern ein HI-END Produkt, was wohl auch erklärt, daß fast alle Betreiber großer Internet-Sites MySQL auf NetBSD einsetzen (NetBSD ist schon länger ein 64 Bit Betriebssystem). Insbesondere für Data Mining ist MySQL hervorragend geeignet. (Glaubt mir wahrscheinlich kaum jemand ;-) Data Mining ist der Versuch, aus riesigen Datenbeständen, z.B. den Aufzeichnungen über Unfälle bei Versicherungen, die Einstufung in die Kasko Versicherung für die einzelnen Autotypen zu ermitteln. Die Verfahren grenzen an schwarze Magie und sind sehr stressend für die Datenbank. Unter MySQL könne diese wochenlangen Analysen dramatisch beschleunigt werden.

Sie können auch Prozeduren in PHP3 schreiben. Lesen Sie hierzu im Kapitel Funktionen in PHP3 für MySQL






17.11 Trigger

Trigger sind Event-Handler, die bei dem Erreichen von Werten in einer Tabelle Stored Procedures oder auch externe Prozesse anstoßen können. MySQL unterstützt keine Trigger. Es ist aber möglich, sich regelmäßig bestimmte Werte in eine ASCII Datei exportieren zu lassen, diese durch ein SHELLScript abzufragen und dann bestimmte Prozesse oder Abfragen zu starten.







17.12 Verschlüsselte Übertragung von Daten

MySQL unterstützt nicht von sich aus die Verschlüsselung von Daten zwischen Client und Server. Es ist aber kein Problem, PPTP auf dem Server zu installieren, und zwar die Point-2 Multipoint Variante von PPTP für UNIX Server. Alle Microsoft Windows Clients unterstützen PPTP von Haus aus, es werden keinerlei Lizenzgebühren fällig. Da die korrekte Installation von PPTP auf dem Server etwas komplizierter ist, möchte ich an dieser Stelle auf das Skript http://www.little-idiot.de/firewall/ oder auf die LINUX VPN HOTO´s verweisen. Damit können Sie eine SQL Datenbank im Internet betreiben, die alle Daten zu Clients verschlüsselt überträgt. Wenn Sie nun noch die Festplatte selber in Echtzeit unter z.B. LINUX mit TCFS verschlüsseln, dann sind Ihre Daten sogar gegen handfesten Diebstahl des Servers selber mit 3DES gesichert. Sicherer geht´s nicht. Wer nach einem VPN Proxy für eine LINUX Firewall sucht - diese ist für Kernel 2.2 mit IPCHAINS frei verfügbar - Siehe http://www.little-idiot.de/firewall/







17.13 Verteilte Datenbanksysteme und Skalierung

Verteilte Datenbanksysteme werden u.a. auch zur Beschleunigung der Abfragen eingesetzt. Hierfür wird eine Software zur Lastverteilung vorgeschaltet. Die wird z.B. bei großen Datenbanken im Internet installiert, deren Leistung bei der hohen Abfragefrequenz schnell zusammenbricht. In diesem Fall kann man entweder über ein PERL Script die Anfragen nach dem ROUND-ROBIN Verfahren (also der Reihe nach) auf die Server verteilen, oder die Software TurboCluster für LINUX installieren, welche die Anfragen automatisch durch den Kernel gesteuert auf die SQL-Server verteilt. Hierbei verbleibt dann nur ein Problem, nämlich die Replikation der Daten bei einer Änderung und die Konsistenz der Datensätze. Während ausgewachsene Datenbanken diese Mechanismen beherrschen (wenn auch mit einigem Programmieraufwand), müssen unter LINUX andere Verfahren gewählt werden. Dem ist ein eigenes Kapitel Verteilte Datenbanken mit einem besonderen PERL Skript gewidmet.







17.14 Datenbank-Replikation

Replikationsmechanismen haben alle mit größeren Problemen zu kämpfen, die durch die lokalen Locks, inkonsistente Datenmengen entstehen. Wer sich einmal andere Datenbanken (z.B. ACCESS) angeschaut hat, der wird feststellen, daß diese alle Felder mit einer sogenannten Replikationskennung versehen, in welcher festgehalten wird, welche Datenbank der Master, welche die replizierte Datenbank ist, und wann welche Einträge verändert wurden. Unter Zuhilfename eines Zusatzfeldes in MySQL kann man dann durch ein Batch-Script die Daten exportieren, die sich z.B. in einer Filiale geändert haben, per DIAL-ON-DEMAND und Filetranfer (FTP) in die Zentrale kopieren und dort in die Master-Datenbank einflechten.

Normalerweise reicht schon der TIMESTAMP Eintrag in einer Tabelle aus, um bei einem Datensatz (Tupel) das Datum und die Uhrzeit der letzten Änderung zu ermitteln.

Das Prinzip ist recht einfach und auch sicher. Der Vorteil ist, daß man keine Standleitung benötigt. Der Nachteil ist, daß man etwas Shellporgrammierung beherrschen muß. Siehe auch Kapitel Verteilte Datenbanken.







17.15 Große Datenbanken > 2 GByte

Große Datenbanken mit vielen Millionen Einträgen mußten bisher immer auf SGI IRIX, DEC ULTRIX / DIGITAL UNIX, MVS, SIEMENS RELIANT oder SOLARIS installiert. Der Grund liegt in der Begrenzung des 32 Bit Adressraumes zur Ansteuerung der Datenbank auf der Festplatte. Damit dies trotzdem möglich ist, wurden die SCSI Devices als sogenanntes **RAW DEVICE** angesteuert. Hierbei hat die SQL Datenbank quasi ohne Filesystem Ihre Daten abgelegt. Dies erfordert eine spezielle Programmierung. **MySQL** kann keine **RAW DEVICES** ansprechen. Somit ist (übrigens auch bei anderen Datenbanken) die maximale Größe einer Datenbank auf 2 Gigabyte begrenzt. Setzt man nun 64 Bit LINUX mit DEC APLHA oder SPARC Prozessoren ein, dann kann LINUX zusammen mit GFS (Giga File System) beliebig große Datenbanken verwalten. Hierzu muß es jedoch nach einigen Anpassungen neu kompiliert werden. Für ODBC-Treiber oder ACCESS als Frontend ändert sich hierbei nichts.







18. Neuigkeiten bei MySQL 3.23

MySQL 3.23 leutet eine neue Generation von MySQL ein, die nun gänzlich den Kunderschuhen entwachsen ist. Das sieht man daran, daß MySQL nun Eigenschaften besitzt, die man nirgendwo sonst finden kann: Welche SQL Datenbank arbeitet schon mit vollen 64 Bit Breite und Datenbankgrößen, die jenseits des physikalisch machbaren liegt ? Viel Wert wurde auch in die weitere Entwicklung in Richtung ANSI SQL 92 gelegt. So werden nun viele wichtige JOIN-Typen unterstützt. Noch mehr Performance durch die nun fast beliebige Skalierbarkeit (Mehrere CPU´s, Multi Threads, 64 Bit...) machen MySQL inzwischen einzigartig. Hier nun die wesentlichen Änderungen der neuen Generation von MySQL 3.23 gegenüber den älteren Versionen:

1. Volle **Multi-Thread** Unterstützung für Kernel-Threads. Das bedeutet, daß **MySQL** mit der Zahl der CPU´s im System skaliert. Das macht sich insbesondere bei komplexen Select-Statements auf Internet-Servern bemerkbar.
2. **C, C++, Eiffel, Perl, PHP, Python und TCP API´s** sind verfügbar. Siehe Kapitel MySQL Client Werkzeuge
3. **MySQL** ist auf beliebige Plattformen bereits portiert. Siehe Kapitel Betriebssysteme für MySQL
4. Viele Typen für Spalten: signed, unsigned integer, 1,2,3,4 und 8 Byte lange Zahlen, FLOAT, DOUBLE, CHAR, VARCHAR, TEXT, BLOB, DATE, TIME, DATETIME, TIMESTAMP, YEAR, SET und ENUM Typen. Siehe Kapitel CREATE
5. Sehr **schnelle JOINS**, da eine optimierte "one-sweep multi-join" Routine implementiert ist.
6. Volle Operatoren und Funktionsunterstützung im **SELECT** und **WHERE** Teil der Abfragen. Beispiel:

```
mysql> SELECT CONCAT(first_name, " ", last_name) FROM tbl_name
           WHERE income/dependents > 10000 AND age > 30;
```

7. SQL Funktionen sind in einer hochoptimierten Klassenbibliothek implementiert und sollten so schnell wie das theoretische Limit der Hardware sein. Normalerweise wird während und nach einer Abfrage kein Speicher angefordert bzw. freigegeben.
8. Unterstützung für SQL **GROUP BY** und **ORDER BY** Ausdrücke. Unterstützung für **Aggregatfunktionen**, wie COUNT(), AVG(), STD(), SUM(), MAX() und MIN().
9. Unterstützung für **LEFT OUTER JOIN** mit ANSI SQL und ODBC Syntax.
10. Es dürfen **Tabellen von verschiedenen Datenbanken** in derselben Abfrage erfolgen (Ab Version 3.22).
11. Das Privilegien und Passwort System ist sehr flexibel und sicher. Es erlaubt **HOST basierte Überprüfug der Abfragen**. **Verschlüsselung** der Passworte ist selbstverständlich.
12. **ODBC** (Open Database Connectivity) für Windows 95/98/NT (mit Quellcode) ist verfügbar. Volle (und mehr) Unterstützung des **ODBC 2.5** Standards von Microsoft. Access Anbindung möglich. Siehe Kapitel ODBC Unterstützung
13. Sehr schnelle **B-TREE** disk-tables mit Indexkompression
14. **16 Indizes pro Tabelle** sind erlaubt. Jeder Index kann aus bis zu 15 Spalten oder Teilen von Spalten bestehen. Die maximale Länge ist 256 Bytes, kann aber durch neue Kompilation erweitert werden. Ein Index kann ein Prefix eines CHAR oder VARCHAR Feldes benutzen.

15. Feste und variable Einträge (Felder) ersparen viele Speicherplatz
16. Speicherresidente **Hashtabellen**, die als **tempöräre Tabellen** abgelegt werden.
17. Unterstützung für **sehr große Datenbanken**. Einige Datenbanken enthalten **50.000.000** Einträge.
18. Alle Spalten besitzen Standard Werte. Der Befehl INSERT, der ein Subset von Spalten in einer Tabelle einfügt, setzt Standard Werte ein, außer wenn explizit angegeben.
19. Unterstützung für GNU automake, autoconf und libtool für bessere Portabilität
20. Geschrieben in C und C++. Unterstützung von einer großen Anzahl von Kompilern
21. **Thread basierte Speicheranforderung** (malloc)
22. **Keine Speicher Lecks**, entsprechend der Analyse mit **Purify**
23. MySQL enthält **isamchk** einem sehr schnellen Werkzeug für die Tabellenüberprüfung, Optimierungen und Reparatur von Tabellen. Siehe Kapitel MySQL installation
24. Support für **ISO-8859-1** Latin1 Format. Alle Vergleiche für normale Spalten von Strings sind case sensitiv.
25. Die Sortierung von Elementen erfolgt nach dem **ISO-8859-1** Latin1 Set. Es ist möglich, dieses im Quellcode zu verändern, um neue Sortierungsreichenfolgen festzulegen. Um ein Beispiel für **fortgeschrittene Sortierung** zu sehen, sollte man sich einmal die Tschechische Sortierung (Czech) anschauen. **MySQL** unterstützt verschiedene Character-Sets, die bei der Kompilierung angegeben werden können.
26. **Aliases** auf Tabellen und Spalten entsprechen dem **SQL92 Standard**
27. DELETE, INSERT, REPLACE und UPDATE geben als Rückgabewert aus, wieviele Reihen von der Änderung betroffen waren.
28. Funktionsnamen dürfen nicht mit einem Tabellen oder Spaltennamen kollidieren. Z.B. ist **ABS** eine gültiger Spaltenname. Die einzige Beschränkung ist, daß keine Leerzeichen zwischen dem Funktionsnamen und dem "(", das folgt, sein dürfen. Siehe Kapitel Ist MySQL empfindlich bei reservierten Worten
29. Alle **MySQL** Programme (Kommandozeile) können mit den Optionen **--help** und **-?** aufgerufen werden. Es wird eine Hilfe ausgegeben.
30. Der **MySQL** Server kann Fehlermeldungen in vielen verschiedenen Sprachen an den Client übergeben. Siehe Kapitel Welche Sprachen sind von MySQL unterstützt ?
31. Clients, die an dem **MySQL** Server angebunden sind, können über TCP/IP Verbindungen, UNIX Sockets, oder **named pipes** unter Windows NT den Server kontaktieren.
32. Das **MySQL** spezifische Kommando **SHOW** kann dazu verwendet werden, um Informationen über die Datenbanken, Tabellen und Indizes abzufragen. Der Befehl **EXPLAIN**, siehe Kapitel EXPLAIN dient dazu, zu bestimmen, wie der Optimizer eine Abfrage durchführt (DEBUGGING)
33. Ein Hauptunterschied zwischen Version 3.23 und den älteren Versionen ist, daß 3.23 eine neue, viel schnellere ISAM Library (MyISAM) besitzt, die an SQL besser angepasst ist.
34. **BIG File Support** für **64 Bit** Architekturen. Wer einmal Video digitalisiert hat, der wird festgestellt haben, daß bei einer Filegröße von 2 Gigabyte bzw. 4 Gigabyte als Ausnahme (je nach Filesystem) Schluß ist. Bisher mußten große Datenbankserver entweder mit einem 64 Bit Filesystem ausgestattet sein, oder die Datenbank hat direkt auf das RAW-DEVICE (meist SCSI) geschrieben, indem es quasi ohne Formatierung direkt in den SCSI Controller geschrieben hat. SGI IRIX, SOLARIS 2.7 können nun von Hause aus viel größere Dateien schreiben. LINUX 2.0/2.2 auf DEC ALPHA und LINUX 2.2 auf (ULTRA) SPARC, sowie NetBSD für DEC ALPHA können nun unter Verwendung von GFS (GIGA FILESYSTEM) ebenfalls riesige Files verwalten. **MySQL** unterstützt nun diese Eigenschaften. Hierzu muß MySQL mit dem GCC Compiler für 64 Bit Architekture mit kleinen Änderungen im Quellcode neu übersetzt werden. Für die Dateigröße können allerdings nur 63 Bit

verwendet werden...also nur halb so viel ;-((

35. **Erweiterte Tabellen Typen**
36. Es können im RAM sog. **HEAP Tabellen** angelegt werden, die extrem schnell arbeiten (ähnlich RAMDISK)
37. Neuer Befehl **LOAD_FILE(filename)**, um sich nur die Inhaltsangabe (Spaltennamen) einer Tabelle anzuschauen.
38. Neuer **Operator** <=>, der genauso, wie = arbeitet, allerdings **TRUE** zurückliefert, wenn beide Argumente NULL sind, also eine Art **XOR** Verknüpfung
39. Unter ODBC 3.0 gibt es nun die Funktion **EXTRACT(interval from datetime)**.
40. Volle Unterstützung von **ODBC 2.5** (Siehe Microsoft)
41. REPLACE ist nun schneller
42. **LIKE** kann nun kleine Unterschiede zwischen Buchstaben, wie é und e als gleich behandeln. Das bedeutet, daß TRUE zurückgegeben wird.
43. **LIKE** kann nun mit **SHOW STATUS** kombiniert werden
44. **SHOW STATUS** gibt nun eine Spalte mit Privilegien aus (privilege)
45. **SHOW INDEX** gibt die Spalten **packed** und **comment** aus
46. Tabellen können nun zur besseren Dokumentation Kommentare enthalten (CREATE TABLE ... COMMENT "Kommentar"
47. CREATE TABLE table_name (col int not null **UNIQUE**) ist nun erlaubt
48. Neue Syntax: **CREATE TABLE SELECT**
49. Neue Syntax: **CREATE TABLE IF NOT EXISTS**
50. Anlegen einer Spalte mit **CHAR(0)** ist nun erlaubt
51. Neue Syntax: **DATE_FORMAT()** erfordert nun ein % vor jedem Formatbuchstaben
52. **DELAYED** ist nun ein **reserviertes Wort**
53. Eine **Beispielprozedur** ist hinzugefügt worden: Siehe hierzu auch Sql_analyse.c . Diese Prozedur wird die Daten in Ihrer Abfrage beschreiben. Beispiel zur Anwendung:

    ```
    SELECT ... FROM ... WHERE ... PROCEDURE ANALYSE([max elements,[max memory]])
    ```

    Diese Prozedur dient als Beispiel für die **Implementierung eigener Prozeduren**,
54. **BINARY CAST** forciert, das Strings case sensitive verglichen werden
55. Neue Option **--skip-show-databases** in **mysqld**
56. Die Überprüfung, ob eine Reihe sich verändert hat, funktioniert nun auch mit BLOB/TEXT Spalten
57. **INNER JOIN** ist nun erlaubt (INNER ist nun ein reserviertes Wort)
58. Die **Zugriffrechte** können sich nun auch neben **Hostnamen** auch auf **Netzmasken** (Subnetze) beziehen.
59. Bei einem Vergleich von **NOT NULL DATE/DATETIME** Spalten, die IS NULL sind, wird nun mit 0 verglichen, um einige **Differenzen mit ODBC auszuschalten.**
60. **NULL IN (...)** gibt nun **NULL** anstelle von 0 zurück. Dies dient dazu, daß null_column NOT IN (...) nicht auf NULL Werte zutrifft.
61. Fehlerkorrektur bei dem Speichern von Fließkommawerten in TIME Spalten
62. **TIME Strings** werden nun präziser analysiert:

```
[[DAYS] [H]H:]MM:]SS[.fraction]
[[[[[H]H]H]H]MM]SS[.fraction]
```

Der zweite Teil des **fraction** Abschnittes wird erkannt und ignoriert

63. **LOAD DATA INFILE** kann nun auch mit **LOW_PRIORITY** versehen werden
64. Der Standard INDEX Name benutzt nun dieselbe Groß/Keinschreibung, wie der Spaltenname
65. Zahl der **maximalen Verbindungen** wird auf **100** festgelegt
66. Größerer **Puffer** bei **LOAD DATA INFILE** (schneller)
67. **DECIMAL(x,y)** arbeitet nun entsprechend dem ANSI SQL Standard
68. **UDF Aggregatfunktionen** wurden hinzugefügt: **LAST_INSERT_ID()** wird nun bei **INSERT INTO ... SELECT** geupdatet.
69. JOINS sind nun schneller
70. **SELECT DISTINCT** ist nun viel schneller, da es die neue UNIQUE Funktion in MyISAM verwendet. Die Ausgabe bei DISTINCT wird nun nich mehr sortiert
71. Temporäre RAM HEAP Tabellen werden nun automatisch in MyIASM Tabellen auf die Festplatte ausgelagert, falls nötig. **table is full error** sollte somit nicht mehr erscheinen....
72. Alle C Client API Makro's sind nun Funktionen, damit die shared libraries zuverlässiger arbeiten. Daher ist z.B. der Aufruf einer Funktion: call mysql_num_fields() bei einem MySQL Feld nicht mehr erlaubt. Stattdessen muß das Makro mysql_field_count() verwendet werden.
73. Benutzung des **LIBWRAP** nun möglich zur Identifizierung von CLients
74. **AUTO_INCREMENT** setzt die Spalten nun automatisch auf NOT NULL
75. Show NULL ist nun der Standardwert für AUTO_INCREMENT Spalten
76. Neues Statement: **SQL_BIG_RESULT** . SQL_SMALL_RESULT ist nun Standard
77. Shared Lib nun im RPM Format verfügbar
78. Option **--enable-large-files** ist nun bei configure möglich
79. Neue Optionen bei CREATE TABLE: **PACK_KEYS** und **CHECKSUM**
80. Neue Option bei mysqld: **--default-table-type**





19. Wie stabil ist MySQL ?

Diese Frage ist natürlich von überragender Bedeutung. Dieser Abschnitt soll einige dieser Fragen klären. Es werden hier daher die typischen Anfragen aus der Mailing-Liste für BUG-Reports beantwortet. Sie werden vielleicht im Folgenden bemerken, daß es kaum von irgendeinem Hersteller so ehrliche Aussagen zu Fehlern in Software gibt. Dank der OpenSource Entwicklung weiß nun der Systemadministrator, an welcher Stelle er zuerst suchen muß, also wo die Wahrscheinlichkeit, den Fehler schnell zu finden, am größten ist. Das steht krass im Gegensatz zur Politik einiger Firmen, die Fehler nur dann zugeben, wenn diese offensichtlich sind, oder Fehler einfach als Standard erklären .....

Auf TCX, arbeitet **MySQL** seit Mitte 1996, also inzwischen 3 Jahre ohne Probleme. Als **MySQL** der Öffentlichkeit zugänglich gemacht wurde, stellte sich heraus, daß einige Teile des Codes noch nicht ausreichend getestet waren. Abfragen, die User in einer etwas anderen Syntax formulierten, führten zu Problemen. Jedes neue Release hatte geenüber dem Vorgänger stets weniger Probleme (auch wenn neue Features hinzukamen).

Bisher jede **MySQL** Version war ohne Einschränkungen einsetzbar. Probleme treten gewöhnlich immer dann auf, wenn User versuchen, neue Befehle aus der "grauen Zone" zu verwenden. Dieser Abschnitt beschränkt sich auch nur auf die bekannten Probleme. Alle bekannten und berichteten Bugs sind in der letzen Version korrigiert worden, mit Ausnahme derjenigen Bugs, die Design-Änderugen betreffen. Siehe hierzu Kapitel Bekannte Fehler und Design - Fehler in MySQL

MySQL ist in verschiedenen Ebenen und verschiedenen, voneinander unabhängigen Modulen implementiert worden. Diese Module sind hier nun genau beschrieben und es wird angegeben, wie gut diese jeweils getestet sind. Der Leser mag dann selber entscheiden, welche Probleme eventuell auf ihn zukommen, in Abhängigkeit der Implemenierung der Datenbank:

- ISAM table handler - stabil. Dieser Handler speichert und liest alle Daten. Es wurde in allen **MySQL** Versionen nicht ein einzige Fehler berichtet bzw. entdeckt. Es gibt auch keine Berichte über Datenverluste.
- Parser und Syntax-Analyser - stabil. Eine Berichte über Probleme seit langer Zeit.
- C Client Code - stabil. Keine bekannten Probleme soweit. In den frühen 3.20 er Versionen gab es einige Begrenzungen in der buffer size, welche aber mit der Version 3.21 beseitigt wurden. Die buffer size wird nun dynamisch bis zu einer Größe von 24 MByte angepasst.
- Standard Client Programme - stabil. Diese sind mysql, mysqladmin, mysqlshow, mysqldump und mysqlimport.
- SQL Funktionen - stabil. Die Basisfunktionen von SQL, String-Klassen und das dynamische Memory Handling laufen ohne Probleme. Kein einziger BUG Report.
- Query optimizer - gamma
- Range optimizer - stabil
- Join optimizer - stabil
- Locking - gamma. Diese Funktion ist sehr vom Betriebssystem abhängig. Auf einigen Systemen gibt es Probleme mit dem Standard Locking (fcntl()). In diesem Fällen muß der

MySQL Dämon mit den Optionen **--skip-locking** kompiliert werden. Die Probleme treten nur auf bei einigen LINUX - und SunOS Systemen, wenn ein NFS-Filesystem verwendet wird.

- Linux threads - gamma. Das Problem läßt sich mit obigen Kompileroptionen beseitigen.
- Solaris 2.5+ pthreads - stabil. Dieses Betriebssystem wird bei TCX eingesetzt zur Entwicklung !
- MIT-pthreads - gamma. Es wurden keien Bugs mehr berichtet seit der Version 3.20.15, als keine Probleme mehr seit der Version 3.20.16. MIT-pthreads sind eventuell ein wenige langsam, weil die ca. 1/20 Sekunde zwischen jeder Abfrage verzögern, aber Index-basierte SELECT Abfragen sollten normalerweise innerhalb eines Zeit-Zyklus stattfinden, sodaß hier keine mutex locking/tread juggling Probleme auftreten sollten. Andere Thread Implementierungen - alpha und beta: Es sind bisher viele Probleme aufgetreten, hauptsächlich mit dem Thread - Implementierungen selber.
- LOAD DATA.., INSERT .... SELECT - stabil. Es traten nur einige Mißverständnisse bezüglich der Syntax auf. Keine Probleme im Code soweit.
- ALTER TABLE - gamma. Kleine Änderungen in der Version 3.22.12
- DBD - gamma. Nun von Jochen Wiedmann betreut.
- mysqlaccess - gamma. Geschrieben und betreut von Jochen Wiedmann
- GRANT - beta. Ein paar Änderungen wurden in 3.22.12 vorgenommen.
- MyODBC (ODBC SDK 2.5) - beta. Mit einigen Programmen scheint es gut zu funktionieren.








20. Unterstütze Betriebssysteme und Bibliotheken

MySQL ist auf eine unglaublich große Anzahl von Systemen portiert worden, um Prinzip läuft MySQL auch auf Handhelds, da es sehr kompakt ist. Für fast alle Prozessoren und Betriebssysteme stehen Binaries zur Verfügung, die sich einfach installieren lassen. Eine Anleitung liegt jeweils bei. Mit Hilfe des GCC Kompilers steht es jedem frei, MySQL selber zu kompilieren. Das original Handbuch gibt hier genauere Hilfe zu speziellen Problemen mit einigen Betriebssystemen. Es gibt jedoch insbesondere bei höheren Leistungsanforderungen einige Aspekte, die genauer betrachtet werden müssen. Diese sind insbesondere unter Microsoft Windows wichtig.







20.1 UNIX und UNIX Derivate

Da **MySQL multithreadding** unterstützt, werden so auch einige Mehrprozessor-Systeme voll ausgenutzt.

MySQL kann auf den folgenden Betriebssystemen mit multithreadding Bibliotheken installiert werden:

- SOLARIS 2.5 & 2.6 mit native Threads
- SunOS 4.x mit dem MIT Threads Paket
- BSDI 2.x, 3.1 und 4.x mit dem MIT Threads Paket
- BSDI 3.0, 3.1 und 4.x mit native Threads
- SGI IRIX 6.x mit native Threads
- AIX 4.x mit native Threads
- DEC UNIX 4.x mit native Threads
- Linux 2.0+ mit LinuxThreads 0.7.1 oder glibc 2.0.7
- FreeBSD 2.x mit dem MIT Thread Paket
- FreeBSD 3.x mit native Threads
- SCO OpenServer mit einem jüngeren Port des FSU-Thread Pakets
- SCO UNIXWARE 7.0.1
- Tru64 UNIX
- NetBSD 1.3 Intel und NetBSD 1.3 Alpha
- Openbsd 2.x mit MIT-pthreads
- HP-UX 10.20 mit dem MIT-pthreads
- HP-UX 11.x mit native threads
- Win95 und NT, keine Threads
- OS/2 Warp 3, FixPack 29 und OS/2 Warp 4, FixPack 4







20.2 MySQL unter Windows 95/98/NT

Die neueste Win95/98/NT-Version ist gegenwärtig als 30 Tage laufende Shareware Version verfügbar. Die Quellcodes werden nur an Entwickler ausgegeben. Die Lizenz kostet 170 Euro zzgl. 170 Euro für E-Mail Support. Unter Windows NT läßt sich der **MySQL** Quellcode der UNIX Variante auch mit dem Compiler **egcs** oder GNU C kompilieren. Den Compiler findet man auf der Site http://www.cygnus.com oder auf einem der Mirror im Internet. **MySQL** verlangt für schnelles Arbeiten ein Betriebssystem mit **preemptive multitasking**, was leider bei Windows 95/98 nicht gegeben ist. **MySQL** läuft unter Microsoft Windows 98 und NT zwar scheinbar auch mit Threads, jedoch sind diese etwas anders implementiert, sodaß man mit erheblich höherem RAM Verbrauch kalkulieren muß. Unter Windows 95 läuft MySQL leider nicht korrekt, da bei jeder Verbindung ein eigener Thread erzeugt wird, der 200 Byte nicht mehr wieder freigibt. Das bedeutet bei 10.000 Zugriffen am Tag, daß 2 MByte an RAM täglich einfach verschwinden. Windows 95/98 hat außerdem noch einige Probleme mit **blocking sockets** und noch einige Probleme im TCP/IP Stack, sodaß hier DoS (Denial Of Service Angriffe) einfach den Datenbankserver stoppen können. Man sollte daher möglichst NT einsetzen. Unter Windows 98/NT gibt es jedoch leider keine UDF´s (User Defined Functions), da der Quellcode für MS Windows nicht frei verfügbar ist....







20.3 Mehrprozessor - Systeme

Dank der MIT/pthreads profitiert MySQL automatisch von mehreren Prozessoren, wenn das Betriebssystem dies auch anbietet. Unter LINUX Kernel 2.2 ist die Ansteuerung von bis zu 4 CPU´s kein Problem. Unter LINUX 2.3 werden dann automatisch bis zu 16 CPU´s angesteuert. Ein preiswertes DUAL Celeron - Board (kein Scherz) ist das Abit Board, welches von Westmall vertrieben wird. CPU-Leistung und RAM können in einer SQL Datenbank nie genügend vorhanden sein, zumal MySQL automatisch im RAM temporäre Tabellen anlegt, und sich stets selber optimiert. **MySQL** ist die schnellste SQL - Datenbank für kleinere einfache Anwendungen auf dem Markt.







20.4 Auswahl von Hardware für MySQL

Eine häufig gestellte Frage ist diejenige nach der ausreichenden Hardware. In der Praxis hat sich herausgestellt, daß RAM durch nichts zu ersetzen ist, außer durch noch mehr RAM. Ein Pentium 75 mit 128 MB RAM unter LINUX ist in der Lage, eine Suche durch 800.000 Einträge in einer Datenbank, die ca. 100Mbyte Größe besitzt, in weniger als 3 Sekunden zu absolvieren. Beim Einsatz von Solaris sollten 64 Mbyte für das Betriebssystem hinzu addiert werden, um eine akzeptable Leistung zu erreichen. Bei Windows NT können erhebliche Performance-Einbrüche bei mehr als 30 simultanen Zugriffen auftreten (keine Threads). Wer mit kleineren Datenbanken bis zu einigen Megabyte Größe arbeitet, der muß sich auch bei einem 80486-DX 40 mit 16 MByte RAM keine Gedanken machen. Grundsätzlich sollte die RAM Ausstattung der Größe der Datenbank entsprechen, das dürfte aber wohl auf alle Betriebssysteme zutreffen. Für den Einsatz im Internet oder bei vielen simultanen Zugriffen sollte der Einsatz von NetBSD, FreeBSD, Solaris oder LINUX (in der Reihenfolge) erwogen werden. NetBSD ist das schnellste System aufgrund seines ausgefeilten Memory Managements, wenn auch mit geringem Abstand. LINUX sollte erst ab der Version 2.2 zum Einsatz kommen. Grund sind diverse Performance - Einbrüche beim Einsatz von **MySQL** mit dem Apache WWW-Server, PHP3 und PERL, die bei NetBSD und Solaris nicht auftraten. Die Abstände der Performance sind aber nur bei hoher Last zu bemerken. Viel wichtiger für die Performance als WWW-Server ist z.B. die Konfiguration des Apache Webservers, der sich durch den Einsatz von MOD_PERL oder MOD_PHP dramatisch beschleunigen läßt. Damit wird auch die Ansteuerung von MySQL viel schneller, so etwa um Faktor 20-40. Es kann dann sein, daß z.B. ein 80486/66 mit 16 MB RAM als Internet-Server doppelt so schnell ist, wie Pentium 450 mit 64 MB, zumindest unter LINUX.







21. Unterschiede zwischen MySQL-Win32 und UNIX

MySQL-Win32 hat sich als sehr stabil erwiesen. Leider gibt es einige Unterschiede zwischen der Windows Version und den UNIX Derivaten, die häufig zu Problemen führen:

1. **Win95 und Threads** Unter Windows 95 gibt es ein Memory Leak, welches pro aktiviertem Thread 200 Bytes von dem Hauptspeicher nach dessen Beendigung nicht mehr freigibt. Das bedeutet, daß nach ca. 5000 Zugriffen ca. 1 MByte RAM Verlust zu beklagen ist. Der Fehler liegt in den Libraries von Windows 95, NT und Windows 98 besitzen diesen Fehler nicht.
2. Windows 98 muß nach ca. 30 Tagen neu gestartet werden, da es einfach aufhört, zu arbeiten.
3. **Blocking Sockets** sorgen bei der Win32 Version von MyQL dafür, daß bei jedem Zugriff eines Clients auf den Port dieser gesperrt wird. Das hat zur Folge, daß eine Verbindung nicht nach einigen Minuten/Stunden nicht automatisch freigegeben (disconnektiert) wird, wie dies unter UNIX der Fall ist. Wenn eine Verbindung hängt, das ist immer dann der Fall, wenn ein Client mal abstürzt. Dieser Thread wird dann nicht automatisch freigegeben und verbraucht dann RAM. Ein DoS Angriff wäre sehr einfach möglich... In diesem Falle muß MySQL von Zeit zu Zeit neu gestartet werden....ebenso, wie Windows NT.....**mysqladmin** mit den Optionen **kill** und **shutdown** funktioniert in diesem Fall nicht.
4. UDF (User Defined Functions) können momentan unter Win32 nicht benutzt werden.
5. **DROP DATABASE** funktioniert immer dann nicht, wenn auf die Datenbank von irgendeinem Client (lesend oder scheibend) zugegriffen wird.
6. MySQL kann nicht mit dem Taskmanager ausgeschaltet werden.
7. Unter Win32 unterstützt MySQL keine User definierten Funktionen.
8. Eine Datenbank, die in Gebrauch ist, kann nicht gelöscht werden.
9. MySQL kann nicht mit dem Taskmanager beendet werden, sondern muß mit dem SHUTDOWN Utility heruntergefahren werden.
10. Die Unterscheidung von Groß/Kleinschreibung bei Dateinamen, Datenbanknamen und Tabellennamen funktioniert nur unter UNIX, nicht unter Win32.
11. Innerhalb eines Statements müssen Groß/Kleinschreibung einheitlich sein. Dieses Statement führt zu Fehlern: **SELECT * FROM my_table WHERE MY_TABLE.col=1;**
12. Unter UNIX ist die Verwendung von "/" einheitlich, unter Win32 müssen alle Charakter als "\" eingegeben werden, so z.B. die Pfadnamen:

```
LOAD DATA INFILE "C:\\tmp\\skr.txt" INTO TABLE skr;
SELECT * FROM skr INTO OUTFILE 'C:/tmp/skr.txt';
```

13. Named Pipes funktionieren unter Win32 nur unter Windows NT. Hierzu muß die Datei c:\my.cnf angelegt werden, und folgende Informationen enthalten:

```
[client]
```

```
host = localhost
```

14. Falls der Fehler **Access denied for user: 'some-user@unknown' to database 'mysql'** erscheint, dann kann der Name des lokalen Hosts nicht korrekt aufgelöst werden. Hierzu muß in der Datei c:\windows\hosts die Zeile **127.0.0.1 localhost** eingetragen werden.






22. Links und Verweise auf andere Server:

In diesem Kapitel befinden sich Link zu anderen WWW-Servern im Internet. Diese Angebote dort unterstehen dem Copyright der jeweiligen Anbieter der Informationen. Diese Linkliste entspricht weitestgehend der Linkliste der englischen Original-Anleitung von MySQL. Darüber hinaus gibt es noch ein Kapitel Beispiele und ausgewählte Links !!!, in welchem außergewöhnlich gute Lösungen zu finden sind. Die Inhalte der Angebote hinter diesen Links sind ausführlich beschrieben.






22.1 Nützliche MySQL-Links

- PHP: Die Server-Side HTML-embedded Scriptsprache, die Alternative zu ASP http://www.php.net
- Ein JDBC Treiber für **MySQL**: http://www.voicenet.com/~zellert/tjFM Der Treiber wird supportet.
- WWW-SQL ist ein CGI-BIN unter PERL, mit welchem man MySQL und PostgreSQL Datenbanken ansprechen kann. Ein komplettes Tutorial ist enthalten. http://www.daa.com.au/~james/www-sql/
- Minivend: Der SHOP mit allem Komfort und natürlich MySQL dahinter. http://www.minivend.com/minivend/
- HeiTML: Eine Serverside Implementierung von HTML and a 4GL Sprache gleichzeitig http://www.heitml.com HeiTML ist kommerziell und ähnlich PHP3, jedoch nicht so gut supportet und auch nicht kostenlos.
- Metahtml: Eine dynamische Sprache für WWW Applikationen http://www.metahtml.com
- VelocityGen für Perl und TCL http://www.binevolve.com ist eine interssante Mischung aus PERL, HTML und TCL. Kommerzieller Support ist gewährleistet.
- Hawkeye Internet Server Suite http://www.hawkeye.net ist sicher eine der fortschrittlichsten Server Suiten, die MAIL, NEWS, CHAT, WWW Protokolle alle in ein z.B: MySQL zusammenführen. Für eine komplexe Siteverwaltung ist HAWKEYE sicher erste Wahl, da die Software auch im Quellcode frei verfügbar ist.
- Network Database Anbindung für Linux und Windows NT http://www.fastflow.com Fastflow ist ähnlich PHP3, leistet auch ähnliches.
- WDB: Web browser als universelles Frontend für fast alle SQL Databanken http://www.wdbi.net/ WDBI ist eine Schnittstelle zwischen PERL einerseits und den SQL DAtenbanken andererseits. Damit kann man SQL Datenbanken beliebig gegeneinander austauschen.
- WebGroove Script: HTML Compiler und Server-side Skriptsprache zum Aufbau von großen Websites. http://www.webgroove.com
- WDBI ist ein WWW-Browser Interface als universelles Frontend, welches **MySQL** voll unterstützt. http://www.wdbi.net
- PERL SQL Interface: Ein einfaches Interface, um beliebige SQL Datenbanken, darunter auch MSSQL, SYBASE..anzusprechen. http://www.perl.org/CPAN/ ist zwar nicht so übersichtlich, aber eine unentbehrliche Resource für alle Interface Programmierer.
- EPERL: EPERL ist ein Apache - Modul, welches wie HeiTML die Einbindung von PERL im HTML Code erlaubt. Hiermit können auch SQL Anweisungen in PERL eingebunden werden. EPERL ist ähnlich EmbedPERL, welches als Modul für Apache exisitert. Leider muß man die PERL Syntax bestehender PERL-Scripte etwas umschreiben, damit diese in HTML eingebettet

werden können, es lassen sich aber alle Module von PERL nutzen, also auch alle Datenbankmodule http://www.engelschall.com

- Dieser Quellcodegenerator erzeugt Visual Basic Code, um auf die MySQL Datenbank zuzugreifen. Der Code eignet sich für Active-X Applets, Anwendungsprogramme, u.s.w. Anschauen ! http://www.pbc.ottawa.on.ca
- Wie man **MySQL** mit Coldfusion auf Solaris/LINUX einsetzt. ftp://ftp.igc.org/pub/myodbc/
- Calistra´s ODBC **MySQL** Administrator. Dieses Toolkit ermöglicht es, über die ODBC Schnittstelle die MySQL Datenbank komplett zu administrieren. Das Toolkit ist ein Windows 98/NT Binary, kann also nach dern Installation des ODBC Treibers zur Administration von MySQL Datenbanken im Intranet und Internet eingesetzt werden. http://calistra.com/MySQL/
- Webmerger: Dieses Werkzeug interpretiert Dateien und erzeugt einen dynamischen Output auf einfachen TAGS basierend. Mit **MySQL** und PostgreSQL über ODBC Treiber. http://www.webmerger.com

---






22.2 Web-Server mit MySQL-Unterstützung

- An Apache authentication module (http://bourbon.netvision.net.il/mysql/mod_auth_mysql/)
- Der Roxen Challenger Web server gehört neben dem Apache-Server zu denjenigen Servern, die ein MySQL Modul für dynamische WWW-Seiten enthalten. Über eine äußerst mächtige und einfach zu erlernende Programmiersprache PIKE kann man dynamische WWW-Seiten, SHOP's u.s.w. erzeugen. http://www.roxen.com







22.3 Beispiele zu SQL Datenbanken

- SQL-Tutorial (in englisch): http://w3.one.net/~jhoffman/sqltut.htm
- http://webdev.berber.co.il/ Hier gibt es nette Beispiele zu MySQL und der Ansteuerung mit PHP3, PERL, ASP. Für Einsteiger eine unentbehliche Resource
- Eine Kontaktdatenbank mit **MySQL** und PHP http://www.webtechniques.com/features/1998/01/note/note.shtml
- Ein SHOP auf **MySQL** basierend. http://www.wse.de
- Ein SHOP mit fast 1 Million Artikeln. http://www.cd-special.de







22.4 Weitere MySQL-Beispiele mit Quellcode

- Little6 Inc, eine JOB-Börse mit MySQL und PHP3 unter LINUX. http://www.little6.com/about/linux/
- DELEC ist ein Werkzeug, um automatisch Table Dokumentationen zu erzeugen. http://www.delec.com/ip/products/prep/examples/BookShelf/index.html
- Steve Fambro benutzt **MySQL** und Webmerger, um 1.2 Millionen Fahrzeugt in UTAH zu speichern. Dank der Indizierung des Kennzeichens ist die Suche superschnell. shredder.elen.utah.edu/steve.html
- World Records ist eine Suchmaschine für Musik, die mit MySQL und PHP läuft. http://www.worldrecords.com
- Beispiele für **MySQL** (siehe Top 10 Liste) http://webdev.berber.co.il
- Eine Kontaktdatenbank mit **MySQL** und PHP. http://www.webtechniques.com/features/1998/01/notes/notes.shtml
- Ein Web-basierter Kalender für größere Gruppen mit PHP. http://modems.rosenet.net/mysql/
- Ein Perl Softwarepaket, um HTML aus einer SQL Tabelle zu generieren, und um SQL Statements aus einem HTML Formular zu generieren. http://www.odbsoft.com/cook/sources.htm
- Einfaches Telefonbuch mit DBI/DBD. http://www.gusnet.cx/proj/telsql/
- **TmySQL**, eine Bibliothek, um **MySQL** mit Inprise (Borland)**DELPHI** zu benutzen. http://www.productivity.org/projects/msql/
- Anleitung zu Installation von PERL Modulen. http://www.iserver.com/support/contrib/perl5/modules.html
- JDBC Beispiele von Daniel K. Schneider. http://tecfa.unige.ch/guides/java/pointers.html
- GNU JDBC Treiber für MySQL http://www.worldserver.com/mm.mysql/
- Postgres SQL Info. http://www.spade.com/linux/howto/
- Object Oriented Concepts Inc. CORBA Anwendungen mit Quellcodes. http://www.ooc.com
- DBWiz, ein Beispiel für Cursor in Visial Basic. http://www.pbc.ottawa.on.ca
- Pluribus ist eine freie Suchmaschine, die lernt, die Qualität Ihrer Ausgaben stetig zu verbessern. Sie merkt sich, welche Links aus der Liste der Suchergebnisse der User bevorzugt. Eine Implementierung mit **MySQL** und PHP. http://keilor.cs.umass.edu/pluribus/
- Stopbit, eine NEWS-Site für **MySQL** und PHP. http://www.stopbit.com
- Beispielskripte bei Jokes2000. http://www.jokes2000.com/scripts/







22.5 Linklisten zu MySQL

- Links zu **MySQL**Datenbanken in Japan/Asien http://www.softagency.co.jp/mysql/index.en.phtml
- Ein kommerzielles Trouble/Problem Tracking System. http://www.open.com.au/products.html
- PTS: Project Tracking System http://www.stonekeep.com/pts/
- Job und Software Tracking System http://tomato.nvgc.vt.edu/~hroberts/mot
- Impementierung des DNS-Servers BIND mit **MySQL** http://www.seawood.org/msql_bind/
- Liste von Web-Providern, die **MySQL** Unterstützung anbieten http://www.wix.com/mysql-hosting
- Volltextsuche mit **MySQL** http://home.wxs.nl/cgi-bin/planeteers/pgidszoek.cgi
- ExportSQL: Ein Script, um die Access Tabellenstruktur in **MySQL** zu konvertieren. http://www.cynergi.net/non-secure/exportsql/
- SAL (Scientific Applications auf Linux) **MySQL** http://SAL.KachinaTech.COM/H/1/MYSQL.html
- **MySQL** Anwendungen und Hilfen Listing http://www.cgishop.com/bin/mysqllist/
- **MySQL*** UDF Registratur http://abattoir.cc.ndsu.nodak.edu/~nem/mysql/udf/







22.6 SQL und Datenbank-Interfaces

- Das JDBC Database Access API http://java.sun.com/products/jdbc/
- **MySQL*** Anbindung an Free Pascal http://tfdec1.fys.kuleuven.ac.be/~michael/fpc-linux/mysql
- Patch für MSQL TCL http://www.gagme.com/mysql
- EasySQL: Ein ODBC ähnlicher Treiber Manager http://www.amsoft.ru/easysql/
- Ein REXX Interface für SQL Datenbanken http://www.lightlink.com/hessling/rexxsql.html
- TCL Interface http://www.binevolve.com/~tdarugar/tcl-sql
- Beispiel für eine Bilddatenbank unter **MySQL** http://tim.desert.net/~tim/imger/
- Der JDBC Klasse 4 Treiber für MySQL: http://doc.trader.dk/mysql/twz-jdbc/jdbc/mysql/docs/twz1jdbcForMysql-doc.html







22.7 ODBC Links

- Der populäre iODBC Treiber Manager ist nun als OpenSource verfügbar http://www.openlinksw.com/iodbc/
- Die FreeODBC Seiten mit allen Informationen, die man über JDBC wissen muß: http://users.ids.net/~bjepson/freeODBC/







23. Installation

Dieses Kapitel beschreibt detailliert die Installation von MySQL unter UNIX bzw. LINUX. Es werden sowohl die Installation der Source-Distribution als auch die Installation der Binary Distributionen (z.B. RPM) beschrieben. Für Einsteiger ist noch eine Diskussion möglicher Fehler, die z.B. bei der Kompilation auftreten können, eingebunden. Falls Ihnen Fehler bekannt sind, die nicht hier aufgeführt sind, bitte ich um Zusendung an stepken@rent-a-databse.de.

Es sollte klargestellt werden, daß es empfehlenswert ist, den MySQL Server stets unter einem Useraccount zu starten. Hierzu muß MySQL neu kompiliert werden, nachdem man das Homeverzeichnis von MySQL in den Kompiler-Optionen angepaßt hat. Wer ganz sicher gehen möchte, der sollte MySQL in einer CHROOT() Umgebung einschließen. Eine ausführliche Anleitung hierzu findet sich auf http://www.little-idiot.de/firewall/, leider noch in englisch.

Für die Installation unter Windows 98/NT sind eigene Kapitel in diesem Handbuch aufgeführt. Dasselbe betrifft auch die Installation von APACHE und PHP bzw. PERL Interfaces.







23.1 Download

Um **MySQL** zu installieren, empfiehlt es sich, die aktuellen Source- und Binary-Distributionen (entsprechend des verwendeten Betriebssystems) aus dem Internet downzuloaden. Bitte nutzen Sie immer diejenigen Mirrors, die Ihnen am nächsten erscheinen !!!!!! Die Software ist überall dieselbe !

Die Mirrors findet man auf der Seite http://www.tcx.se im Unterpunkt **Downloads**, wobei Sie schon aus Geschwindigkeitsgründen die Mirrors nutzen sollten. Die Distributionen liegen in der Form *.tar.gz vor z.B.mysql-3.x.x.tar.gz, wobei beim Download eine Umbenennung in (bei genanntem Beispiel) z.B. mysql-3_x_x_tar.tar stattfindet. Die Dateien müssen vor dem auspacken wieder in die ursprüngliche Form gebracht, d.h umbenannt werden. Hierzu ist der Befehl **mv** (move) zu verwenden (mv 'alter Name' 'neuer Name') Und zwar **_** zu **.** und **tar.tar** zu **tar.gz**, im Beispiel also mysql-3_x_x_tar.tar zu mysql-3.x.x.tar.gz. Nun können sie mit gzip entpackt werden. Die DBI- und Perl-Module können ebenfalls von dieser Internet-Seite bezogen werden.







23.2 Installations-Verzeichnisse

Bei der Installation einer Binär- oder Source-Distribution werden standardmäßig folgende Verzeichnisse angelegt:

- Binärdistribution:

```
/usr/local/mysql/
./bin            Clientprogramme und der mysqld - Server
./data           Logfiles und Datenbanken
./include        include header Dateien
./lib            Libraries
./scripts        mysql_install_db
./share/mysql    Fehlermeldungen
./sql-bench      Benchmarks
```

- Source-Distribution:

```
/usr/local/mysql/
./bin            Clientprogramme und Scripte
./include/mysql include header Dateien
./info           Dokumentation im Info-Format
./lib/mysql      Libraries
./libexec        mysqld-Server
./share/mysql    Fehlermeldungen
./sql-bench      Benchmarks
./var            Logfiles und Datenbanken
```

Die Source und Binär-Distribution unterscheiden sich etwas in den Verzeichnissen, in denen die Binärdateien abgelegt werden:

- Der **mysqld** - Server ist im Verzeichnis **libexec** abgelegt, ein Unterschied zu dem **bin** Verzeichnis.
- Das Verzeichnis für die Datenbanken selber findet sich eher im Verzeichnis **var**, als in **data** .
- Das Vereichnis **mysql_install_db** findet sich eher unter **/usr/local/bin/**, als im Verzeichnis **/usr/local/mysql/scripts**
- Die Header Datei und die Library Verzeichnisse finden sich eher im Verzeichnis **include/mysql** und **lib/mysql**, als in den Verzeichnissen **include** und **lib**







23.3 Kompilieren von MySQL

Zur Installation sind einige Programme erforderlich, die gewöhnlich unter UNIX beiliegen. Wer stolzer Besitzer von SOLARIS 2.5/2.6/2.7 ist, der wird sicherlich einige der unten angegebene Pakete vermissen. Er sollte diese von der Site http://www.sunfreeware.com laden und installieren (wie oben angegeben). Es werden grundsätzlich benötigt:

- **GNU AUTOCONF**
- **GNU GZIP** zum entpacken der Distribution. Hier sollte man stets das unkomprimierte gzip Binary installieren.
- Das Archiv-Programm **TAR**, um die Distribution auszupacken. GNU **tar** sollte auf alle Fälle funktionieren. TAR liegt aber allen UNIX Derivaten normalerweise serienmäßig bei.
- Ein funktionierender ANSI **C(++)** Compiler. **gcc 2.7+** oder **EGCS**, **SGI C++**, **SunPro C++** sind einige von den Compilern, die als funktionierend bekannt sind. **Libg++** wird nicht benötigt, wenn **gcc** verwendet wird.
- Ein **make**-Programm. Es solle, um Probleme zu vermeiden, GNU **make** verwendet werden.
- **GNU FLEX**
- **GNU YACC**
- Alternativ kann statt **YACC** auch **GNU BISON** verwendet werden. Dieses Programm ersetzt **FLEX** und **YACC** zusammen und wird zur Generierung des **MySQL** Befehlsparsers in die Programmiersprache **C** benötigt.

Zusätzliche Installations - Hinweise für SOLARIS und LINUX:

Das Paket GNU **autoconf** ist ein Paket, welches die Anpassung der Quellcodes an die jeweiligen Gegebenheiten des Betriebsystems stark vereinfacht. **autoconf** analysiert die Möglichkeiten des Betriebssystems, fragt nach vorhandenen Bibliotheken und Verzeichnissen und stellt den Compiler so ein, daß **MySQL** die bestmögliche Performance auf dem System erreichen kann. Das gilt insbesondere für Mehrprozessor - Systeme. Für SOLARIS 2.5/2.6/2.7 finden sich alle zum Betrieb benötigten Binaries oder Quellcodes, sowie alle benötigten Programmiersprachen, wie dem C-Compiler und alle weiteren Konfigurationswerkzeuge auf dem Server http://www.sunfreeware.com im Internet. Die Pakete sind leicht mit dem Befehl **pkgadd -d xyz.pkg** zu installieren. Eine ausführliche Anleitung zur Installation findet sich ebenfalls auf dem Server. Man sollte auch die BASH installieren, damit man auf den Vervollständigungs-Modus bei der Eingabe von Verzeichnisnamen (TAB-Taste) und die Befehlswiederholung (Pfeil oben) nicht verzichten muß. Die BASH befindet sich in /usr/local/bin . In die Datei .bash_profile im Homeverzeichnis eines angelegten Users (nicht root) sollten der Suchpfad nach /usr/local/bin/ hinzugeführt werden. (cat >> .bash_profile und dann die Eingabe: export PATH=$PATH:/usr/local/bin/) und die Änderung in der Datei /etc/passwd : csh -> bash sollten genügen, um sich als LINUX User auf Solaris heimisch zu fühlen....:))

LINUX Binaries oder Quellcodes finden sich entweder auf dem Homeserver im Internet oder auf einem der vielen Mirrors. Damit Internet Resourcen gespart werden, sollte die Software stets der am

nächsten liegende Mirror geladen werden. Die Mirrors sind stets aktuell. Da die meisten Distributionen mit dem RPM Format arbeiten, sei hier nur kurz die Installation des **.rpm** Archiv-Files beschrieben: **rpm -i xy.rpm** installiert das Programm. Damit LINUX als Mehrprozessor-System die Fähigkeiten des **multithreadding** nutzen kann, ist zuvor die Installation der entsprechenden Bilbiotheken erforderlich (MIT Threads). Diese sind stets bei allen neueren Distributionen enthalten.






24. Installation der Source-Distribution Schritt für Schritt

Um eine Source-Distribution zu installieren, sollte man folgende Schritte durchführen und danach die Beschreibung im Abschnitt Post-Installation ausführen. Unter LINUX existieren für viele Distributionen, allen voran für die RedHat Distribution eine RPM-Datei. Diese muß nur mit dem Befehl: **rpm -i mysql-VERSION-linux.rpm** installiert werden. Nun zurück zu der etwas mühseligeren Installation der Source Distribution

- Ein Verzeichnis erstellen, in welches die Distribution entpackt werden soll und in dieses wechseln.
- Die **MySQL**-Distribution von einem der **MySQL**-Servern beziehen. Das **.tar-Archiv** sollte einen Namen in der Art **mysql-3.xx.tar.gz** haben
- Die Distribution im Verzeichnis entpacken: **gunzip mysql-VERSION.tar.gz** und anschließend **tar -xvf myssql-VERSION.tar** entpackt die Distribution im aktuellen Verzeichnis.
- In das erstellte Distributions-Verzeichnis wechseln. **cd mysql-3.x.x**
- Die Distribution konfigurieren und kompilieren.Mit **./configure**, anschließend **make** Bei dem Konfigurieren der Releases können ggf. weitere Optionen angegeben werden. Eine Liste der möglichen configure-Optionen kann mit **./configure --help** angezeigt werden.
- Nachdem die Kompilierung erfolgreich ausgeführt wurde, kann die Distribution installiert werden. **make install**

```
mkdir: cannot make directory /usr/local/include/mysql': Permission denied
make[1]: *** [install-pkgincludeHEADERS] Error 1
make[1]: Leaving directory /home/user02/mysql-3.22.16a-gamma/readline'
make: *** [install-recursive] Error 1
```

Sollte diese Meldung erscheinen konnte das Verzeichnis **/usr/local/include/mysql** nicht angelegt werden, da der Benutzer hierfür keine Zugriffsrechte besitzt. Es sind Super-User-Rechte erforderlich.

Die Installation des Perl **DBI/DBD**-Interface wird später beschrieben. Um **MySQL** beim Systemstart automatisch zu starten, ist support-files/mysql.server in das Verzeichnis, in welchem die Startup-Scripts abgelegt werden, zu kopieren (dies ist nicht zu empfehlen). Unter LINUX ist dies das Verzeichnis **/etc/rc.d**. Danach muß ein Eintrag in die Datei /etc/rc.d/rc.local erfolgen.(Bei älteren LINUX Distributionen siehe Handbuch)






24.1 Konfigurations-Schalter:

Diese Einstellungen sind optional.

- **--without-server:** Um nur den Client zu compilieren: **-- without-server**. Wird nur die Client-Library benötigt und es steht kein C++ Compiler zur Verfügung kann in configure der Code, welcher auf einen C++-Compiler testet, entfernt und dann alles mit der Option **--without-server** kompiliert werden. Hierbei können alle Warnungen vom **mysql.cc** ignoriert werden.
- **--prefix:** Sollen die Daten nicht unter **/usr/local/var** abgelegt werden, kann dies mit mit den Optionen: **--prefix=/usr/local** oder **-- prefix=/usr/local --localstatdir=/usr/local/data/mysql** angepasst werden.
- **--with-unix-socket-path:** Sollen die Sockets nicht im Default-Verzeichnis installiert werden, kann dies mittels **--with-unix-socket-path=absolute_file_name** eingestellt werden (Default ist /tmp oder /var/run) .
- **--with-low-memory:** Sollten bei der Compilierung Fehlermeldungen auftreten wie zum Beispiel: **Internal compiler error: program cc1plus got fatal signal 11** oder **Out of virtual memory**sollte die Option **--with-low-memory** übergeben werden. Dies fügt **-fno-inline** für das Kompilieren hinzu. Das Problem liegt darin, daß ungefähr 180MB zum Kompilieren von **sql_yacc.cc** mit Inline-Functions benötigt werden.
- **statische Kompilierung:** Wenn man die Programme statisch kompilieren möchte, um z.B. eine Binary-Distribution zu erstellen, sollte die Option **LDFLAGS=static ./configure** eingestellt werden. Dies umgeht Probleme, die bei einer Veränderung der Bibliotheken auftreten können, wie z.B. nach einem Update der glibc unter LINUX, oder nach dem Einspielen eines Security Patches. Wer beabsichtigt, mehrere SQL Datenbanken auf einem Server zu installieren, der sollte dynamisch linken, um von den **shared libraries** zu profitieren. Hier teilen sich mehrere SQL Datenbanken den Code. Das erspart viel Memory.
- **GCC:** Wenn gcc benutzt wird und keine g++ oder libstdc++ installiert ist, kann dies mittels **CC=gcc CXX=gcc. /configure** konfiguriert werden.
- **Default-Werte:** Wenn keine **DEFAULT**-Werte verwendet werden sollen oder keine Fehler bei **INSERTS** erzeugt werden soll, welche nicht alle **NOT NULL** Tabellenfelder abdeckt, sollte dies mittels **CXXFLAGS=-DDONT_USE_DEFAULT_FIELDS ./configure** konfiguriert werden.
- **Standard Zeichensätze (--with-charset):** Als Default verwendet **MySQL** den **ISO8859-1** Latin1 Zeichensatz. Dieser kann bei der Compilierung mittels **--with-charset=charset** angepaßt werden. Mögliche **charset** sind hierbei:

```
Big5
Czech
dec8
Dos
german1
```

```
hp8
koi8_ru
latin1
latin2
swe7
usa7
Ujis
Sjis
```

Wenn verschiedene Zeichensätze zwischen dem Server und dem Client konvertiert werden sollen, sollte man sich hierzu **SET OPTION CHARACTER SET**näher anschauen. **Achtung:** Wenn der Zeichensatz nach dem Erstellen einer Tabelle gewechselt werden sollte, ist es notwendig den Befehl **isamchk -r -q** auf jede Tabelle anzuwenden. Andernfalls kann es vorkommen, daß die Tabellen nicht nach dem eingestellten Zeichensatz sortiert werden. Computer/OS-spezifische Schalter werden im Original-Manual beschrieben. Falls einige Zeichensätze beim Einsatz als Internet-Datenbank im Browser nicht korrekt erscheinen, so könnte dies an der Zeichensatz Einstellung im Browser selber, fehlerhaftem HTML-Code (Siehe HTML Anleitung von Stefan Münzinger, SELFHTML auf http://www.netzwelt.com/selfhtml/), dem verwendeten Zeichensatz beim Import der Daten in die **MySQL** Datenbank, oder einfach an der falschen Einstellung des verwendeten Zeichensatzes in PERL (man locale).






24.2 Konvertierung von Zeichensätzen

Die Konvertierung von Zeichensätzen aller Art, insbesondere von Dateien im 7 Bit DOS Format in das UNICODE Format oder 8 Bit Format, welches am Ende der Zeile unter UNIX einem die ^M beschert, lassen sich mit dem Programm **recode** durchführen. Dieses liegt einigen LINUX Derivaten bei. Es ist aber auch frei im Internet verfügbar. Die Syntax ist: **recode ibmpc:latin1 datei**. Mit **recode latin1:ibmpc datei** werden die Formate in rückwärtiger Richtung konvertiert. Für den Einsatz in Deutschland sollte **latin1** überall Standard sein.

Die Anpassung unter **MySQL** hat durch die korrekte Auswahl des Datentyps zu erfolgen.







24.3 Einspielen eines Patches

Gelegentlich werden verschiedene Patches in der Mailingliste veröffentlicht. Um diese in die vorliegende Source-Distribution einzubinden, muß man in das Quellcode Verzeichnis von mysql-3.xy zu wechseln, und folgendes eingeben:

```
gunzip < patch-mysql-3.xy.gz
patch -p1 patch-mysql-3.xy
rm config.cache
make clean
make install
```

Danach muß die Distribution wieder neu kompiliert und der Server neu gestartet werden.







24.4 Anmerkungen zum Compilieren

Alle **MySQL** Programme wurden ohne Probleme (keine Warnungen) auf Solaris (unter der Verwendung von gcc) erfolgreich kompiliert. Warnungen können auf Grund von verschiedenen Include-Dateien auftreten. Unter Umständen muß zur Kompilierung von **sql_yacc.yy** der Parsergenerator **bison** verwendet werden.

Sollte ein Fehler der Art:

sql_yacc.yy, line xxx fatal: default action causes potential... auftreten, so ist auf alle Fälle **bison** zu verwenden (GNU yacc). Normalerweise wird dieser nicht benötigt, da **MySQL** mit einer kompilierten **sql_yacc.cc** ausgeliefert wird.

Sollte man statisch gelinkten Code erzeugen wollen, muß an den gcc folgendes übergeben werden: **LDFLAGS=-static ./configure ...**







24.5 Anmerkungen zu MIT-pthreads - FreeBSD

Wenn das verwendete Betriebssystem keinen Native-Threads unterstützt, muß **MySQL** unter Verwenung des **MIT-pthreads Package** kompiliert werden. Dies trifft auf die meisten FreeBSD Systeme, SunOS 4.x, Solaris 2.4 und früher und einige Andere zu. Auf den meisten Betriebssystemen kann die Verwendung der **MIT-pthreads** durch die configure-Option **-with-mit-threads** erzwungen werden. **MIT-pthreads** unterstützt nicht das AF_UNIX Protokoll, welches für die Implementierung der UNIX-Sockets verwendet wird. Deßhalb müssen alle Verbindungen über TCP/IP erfolgen, was etwas langsamer sein kann.




24.6 Anmerkungen zu Perl

Es könnte folgender Fehler auftreten (mysqlperl oder DBD-mysql):

```
/usr/bin/perl: cant resolve symbol __moddi3
/usr/bin/perl: cant resolve symbol __divdi3
```

Dann wurde vermutlich eine ältere Binary-Datei verwendet, welche mit gcc compiliert wurde. In diesem Fall sollte man **-L/usr/lib/gcc-lib/... -lgcc** dem Link Kommando übergeben, wenn die **mysql.so-Library** gebaut wird. **-L/..**< zeigt auf den Pfad in dem sich **libgcc.a** befindet. Ein weiteres Problem kann entstehen, wenn entweder **MySQL** oder Perl nicht mit gcc compiliert wurden. Um dieses Problem zu beseitigen sollten beide Distributionen mittels gcc compiliert werden. Sollte man die Perl-Module auf einem System verwenden wollen, welches kein dynamisches Linken unterstützt (z.B.: SCO), kann man immer noch eine statische Version von Perl mit **DBI** und **DBD-mysql** erzeugen:

- Zuerst wird hierfür eine statisch gelinkte Version von **DBI** erzeugt:

  ```
  perl Makefile.PL LINKTYPE=static
  make
  make install
  make perl
  ```

- Danach muß die neu erzeugte Perl-Version installiert werden. Die Ausgabe von **make perl** enthält das genaue make-Kommandp. Bei **SCO** ist dies

  ```
  make -f Makefile.aperl inst_perl MAP_TARGET=perl
  ```

  Danach wird ein statisch gelinktes **DBD::mysql** erzeugt:

  ```
  perl Makefile.PL LINKTYPE=static
  make
  make install
  make perl
  ```

  Hier sollte auch das neu erzeugte Perl installiert werden (siehe oben).

24.7 Anmerkungen zu bestimmten System/OS-Kombinationen

Dieser Abschnitt ist sehr unvollständig, da mir nicht alle Betriebssysteme zur Verfügung stehen. Hilfe ist hier sehr willkommen!

- **Solaris**

  Solaris tar unterstützt keine langen Dateinamen, weshalb es nicht möglich ist die Distribution zu entpacken. Aus diesem Grund ist es notwendig **GNU-tar** zu verwenden. Eine Vorcompilierte Version für Solaris kann unter **http://www.tcx.se/Downloads** bezogen werden. Sun Native-Threads funktionieren erst ab Version 2.5 oder größer, weshalb für frühere Versionen das MIT-pthread Paket verwendet werden muß. Wenn man den Sun Workshop 4.2 Compiler verwendet kann **./configure** wie folgt verwendet werden:

  ```
  CC=cc CFLAGS=-Xa -fast -xstrconst -mt \
  CXX=CC CXXFLAGS=-xsb -noex -fast -mt \
  ./configure
  ```

  Ferner muß das **configure**-Script editiert werden. Hierbei muß die Zeile:

  ```
  #if !defined(__STDC__) || __STDC__ != 1
  ```

  nach

  ```
  #if !defined(__STDC__)
  ```

  geändert werden.
- **SunOS 4**
- **Linux - alle Versionen**
  - Bei der Verwendung unter Linux sollte das **--skip-locking** Flag verwendet werden, wenn man mysqld selbst starten möchten. Normalerweise wird der Daemon mittels **safe_mysqld** gestartet und das Flag wird automatisch gesetzt. Dieses Vorgehen ist auf grund eines Fehlers in dem Linux File **Locking Call** notwendig. Dieser Fehler existiert bis Linux Kernel 2.0.33.
  - Kann mysqld nicht gestartet werden oder mysql_install_db wird nicht ausgeführt liegt das Problem vermutlich bei den Linuxthreads oder der libc/glibc Library. Hierfür gibt es einige Work-Arounds: Die einfachste Möglichkeit ist die Verwendung der Binary-Version von **MySQL**; nicht die RPM-Version. Ein weiterer Aspekt der für die Verwendung dieser Version spricht ist, daß diese ca. 10 % schneller als jede selbst compilierte Version ist.
  - Wird LinuxThreads verwendet sind mindestens drei Prozesse am Laufen. Diese sind einzelne Threads: einer für den Linux Threadmanager, einer für das Handling von Verbindungen und einer für das Handling von Alarmen und Signalen.
  - Bei der Verwendung von RedHat kann es zu Fehlermeldungen der Art:

    ```
    /usr/bin/perl is needed...
    /usr/sh is needed...
    /usr/sh is needed...
    ```

    kommen. Hierbei ist rpm selbst auf rpm-2.4.11-1.i386.rpm sowie rpm-devel-2.4.11-1.i386.rpm oder höher zu updaten.
- **Linux-x86**
  - Vor der Konfiguration von **MySQL** sollten die LinuxThreads installiert sein. **MySQL** benötigt libc 5.4.12 oder größer. glibc Version 2.0.6 oder größer sollte ebenfalls funktionieren. Es gab diverse Probleme mit den glibc-RPMs. Wenn hierbei also Probleme auftreten, sollten vorhandene Updates installiert werden.
  - Bei manchen älteren Distributionen erzeugt **configure** einen Fehler der Art:

    ```
    Syntax error in sched.h. Change _P to __P in the
    /usr/include/sched.h file.
    See the Installation chapter in the Reference Manual
    ```

    In diesem Fall sollte wie beschrieben zu dem **_P** Macro ein zusätzlicher Unterstrich angefügt werden **__P**.
  - Folgende Warnungen beim Compilieren können ignoriert werden:

```
mysqld.cc -o objs-thread/mysqld.o
mysqld.cc: In function void init_signals():
mysqld.cc:315: warning: assignment of negative value -1 to long unsigned int
mysqld.cc: In function Void * signal_hand(void *):
mysqld.cc:346: warning: assignment of negative value -1 to long unsigned int
```

- Wenn man unter **Debian GNU/Linux MySQL** automatisch starten möchte, sollten folgende Einstellungen vorgenommen werden:

```
cp scripts/mysql.server /etc/init.d/mysql.server
/usr/sbin/update-rc.d mysql.server defaults 99
```

- Wenn mysqld bei Start einen **core dump** erzeugt, kann dies daran liegen, daß eine alte **/lib/libc.a** installiert ist. Diese sollte dann umbenannt werden, sql/mysqld gelöscht und ein neues **make install** durchgeführt werden. Diese Probleme traten bei einigen Slackware-Installationen auf. Bei RedHat 5.0 kann ein ähnliches Problem auftreten, bei manchen glibc-Versionen ebenfalls.

- **RedHat 5.0**
  - Um **MySQL** unter RedHat verwenden zu können muß midestens glibc-2.0.6-9 und glibc-devl-2.0.6-9 installiert sein, da die glibc 2.0.5 einen Fehler bei pthread_key_create enthält. Wird die glibc 2.0.5 verwendet, muß die statisch gelinkte Binary-Distribution von **MySQL** verwendet werden.
  - Will man **MySQL** selbst compilieren wird die aktuelle Version der LinuxThreads benötigt. Die Version steht unter http://www.tcx.se/Downloads/Linux bereit.
  - Wird eine falsche glibc oder LinuxThreads verwendet, führt dies dazu, daß **MySQL** nach jeder Verbindung abstürzt. Zum Beispiel würde **MySQL** nach dem Befehl **mysqladmin version** abstürzen.
  - Ein anderes Symptom ist das **MySQL** nach dem Start abstürzt. Dies kann ggf. behoben werden durch die Option: **LDFLAGS=-static ./configure** Bei manchen RedHat 5.0 - Systemen funktioniert dies wiederum nur ohne die Option: **LDFLAGS=-static** Es ist bekannt, daß dieser Fehler auch mit neueren Versionen der glibc Auftritt (z.B. glibc 2.0.7-4). Für die Quell-Distribution von glibc 2.0.7 steht ein Patch unter http://www.tcx.se/Download/Linux/glibc-2.0.5-total-patch.tgz zur Verfügung. Wenn danach immer noch Abstürze auftreten, sollten man sich die aktuelle Binary-Version von **MySQL** downloaden. Diese ist statisch compiliert und sollte auf allen Linux-Systemen laufen.
  - Um **MySQL** zu debuggen, sollte die aktuelle Dokumentation des Original-Referenz-Manuals zu Rate gezogen werden, da dieser Bereich noch nicht als Übersetzung vorliegt.
- **Linux-Sparc**
  - Bei einigen Implementationen weist **readdir_r** einen Fehler auf. Dies zeigt sich dadurch, daß bei dem Befehl **SHOW DATABASES** immer ein leerer Ergebnissatz zurückgegeben wird. Dies kann behoben werden, indem **HAVE_READDIR_R** in **config.h** entfernt wird.
  - Andere Probleme können durch patchen der Linux-Installation behoben werden. Unter:**http://www.tcx.se/patch/Linux-sparc-2.0.30.diff** steht der entsprechende Patch zur Verfügung. Dieser Patch ist für die Linux- Distribution **sparclinux-2.0.30.tgz.gz**, welche auf **vger.rutgers.edu** bereitgestellt wird. Ferner wird LinuxThreads 0.6 oder höher benötigt.
- **Linux-Alpha**

  Das Hauptprobleme hierbei sind die LinuxThreads. Diese müssen erst gepatched werden, da die RedHat-Distribution eine alte, fehlerhafte LinuxThreads-Implementation verwendet. Hier ist das Vorgehen:
  - Die Sourcen der glibc2.5c von einer GNU Ftp-Site laden.
  - Das File: ftp://www.tcx.se/pub/mysql/linux/patched-glibc-linuxthreads-0.6.tgz laden, und in das glibc ./linuxthreads - Verzeichnis kopieren.
  - glibc konfigurieren und compilieren, aber nicht installieren!
  - Die alte Version von **/usr/lib/libpthread.a** nach **/usr/lib/libpthread.a-old umbenennen.**
  - Die Datei **glibc.../linuxthreads/libpthread.a** nach **/usr/lib kopieren**
  - **MySQL** konfigurieren mittels: **CC=gcc CCFLAGS=-Dalpha_linux_port CXX=gcc CXXFLAGS=-O3 -Dalpha_linux_port ./configure --prefix=/usr/local/mysql** (alles in einer Zeile!)
  - Danach versuchen, ob sich mysys/thr_lock und mysys/thr_alarm ohne Probleme compilieren läßt und ob es funktioniert.
  - mysqld neu compilieren

  Die Details einer Installation auf folgenden Betriebssytemen ist im Original Handbuch auf der Site http://www.tcx.se beschrieben.
- **Alpha-DEC-Unix**

- **Alpha-DEC-OSF1**
- **SGI-IRIX**
- **FreeBSD**
- **FreeBSD 3.0**
- **BSD/OS 2.X**
- **BSD/OS 3.X**
- **SCO**
- **IBM-AIX**
- **HP-UX**

  Für eine HP 9000/7xx/8xx mit HP-UX 9.x/10.x können die Binaries von MySQL aus dem Softwarearchiv von HP bezogen werden. Um MySQL zu installieren, gehen Sie wie folgt vor:

  ```
  shell> /usr/sbin/swinstall -s /path/to/depot mysql.full
  ```

  Danach befinden sich die Binaries in /opt/mysql und es wurden die Startup-Dateien im Verzeichnis **/sbin/init.d** sowie **/sbin/rc2.d** abgelegt.
- **Win32**






25. Installation der Binary-Distribution von MySQL

- Verzeichnis auswählen, in welchem **MySQL** installiert werden soll. Im Folgenden wird die Distribution im Verzeichnis **/usr/local/** entpackt und **/usr/local/mysql** als Installationsverzeichnis verwendet.
- Distribution von einem der in Anhang A aufgeführten Sites beziehen. Zum Beispiel: **mysql-version-OS.tgz MySQL**-Binary-Distributionen werden als gepackte tar-Archive vertrieben.
- Die folgenden Kommandos setzten Schreibrechte im Verzeichnis **/usr/local** voraus.

```
cd /usr/local
gunzip mysql-VERSION-OS.tar.gz
tar -xvf mysql-Version_OS.tar
ln -s mysql-VERSION-OS mysql
```

  Das erste Kommando erzeugt ein Verzeichnis mit dem Namen **mysql-VERSION-OS** in welchem die Distribution entpackt wird und das Zweite setzt einen symbolischen Link auf dieses Verzeichnis.
- Wechseln in das Installationsverzeichnis:

```
cd mysql
```

  Hier finden sich weitere Dateien und Unterverzeichnisse. Die wichtigsten für die Installation sind
  - **./bin:** enthält Clientprogramme und Server. Dieses Verzeichnis sollte in der PATH-Variablen eingetragen werden.
  - **./scripts:** Hier liegt das Script **mysql_install_db** welches zur Initialisierung des Servers benötigt wird.







25.1 PERL installieren

```
cd perl/DBI
perl Makefile.PL
make
make install
cd ../Mysql-modules
perl Makefile.PL
make
make install
```




25.2 Starten des Servers

Zum Start des Servers sollte immer das Script **./safe_mysqld** verwendet werden. **./safe_mysqld** erwartet eine der beiden Konstellationen als **WAHR:**

```
das Script wird direkt aus dem Installationsverzeichnis gestartet
(z.B. /usr/local/bin/
oder
Der Server wurde unter /my/ installiert.
```

Um eine korrekte Ausführung des Scripts zu gewährleisten sollte man ein **cd**nach **/usr/local/bin/** ausführen und danach **./safe_mysqld** ausführen oder das Script entsprechend anpassen, sodaß es **/usr/local/bin/** statt **/my** als Basis-Verzeichnis annimmt.

Kann mysqld die erforderliche Datenbank oder das entsprechende Verzeichnis nicht finden, erscheint folgende Fehlermeldung.

```
Starting mysqld daemon with databases from /usr/local/var
./safe_mysqld: /usr/local/var/tunix.err: No such file or directory
./safe_mysqld: /usr/local/var/tunix.err: No such file or directory
./safe_mysqld: /usr/local/var/tunix.err: No such file or directory
mysqld daemon ended
```

Bei der Ausführung von

```
bin/mysql --help
```

werden alle Optionen aufgelistet, welche mysqld mit übergeben werden können sowie alle aktuellen Pfadangaben. Normalerweise benötigt man hier nur die Verwendungen von

```
--basedir=path
```

Die path-Option kann getestet werden mittels:

```
bin/mysql --basedir=/usr/local --help
```

Zur Verwendung von **mysqlaccess** mit einer nicht im Standardverzeichnis installierten Version von **MySQL** muß im Script **bin/mysqlaccess** in Zeile 308 der Pfad angepaßt werden:

```
$MYSQL = /usr/local/bin/mysql --batch --unbuffered;
```

Wird dies nicht durchgeführt, kommt es zu der Fehlermeldung: **broken pipe** bei der Ausführung von mysqlaccess.







25.3 LINUX RPM-Installation

Unter LINUX sollte man die Distribution als RPM-File installieren.

```
shell> rpm -i MySQL-VERSION.i386.rpm MySQL-client-VERSION.i386.rpm
```

Oder um nur den Client zu installieren:

```
shell> rpm -i MySQL-client-VERSION.i386.rpm
```

RPM plaziert die Daten in das Verzeichnis **/var/lib/mysql** und erstellt im Verzeichnis **/sbin/rc.d** bzw. bei LINUX 2.2 im Verzeichnis **/etc/rc.d/** die Startup-Dateien.







25.4 Post-Installation

- Nachdem **MySQL** installiert wurde muß das Sicherheits/Zugriffssystem initialisiert und die Funktion getestet werden. Ferner kann festgelegt werden, ob der Server automatisch gestartet und gestoppt werden soll. In der Beschreibung bezieht sich **BINDIR** auf das Verzeichnis in welchem **mysqladmin**, **safe_mysqld** etc. installiert wurden. Bei einer Binary-Distribution ist dies das **./bin**-Verzeichnis der Distribution, bei einer Source-Distribution ist dies **/usr/local/bin**, sofern kein anderes Verzeichnis bei **./configure** angegeben wurde. **EXECDIR** bezieht sich auf das Verzeichnis, in welchem der Server **mysqld** installiert wurde. Bei einer Binary-Distribution ist dies gleich dem **BINDIR**-Verzeichnis bei einer Source-Distribution ist dies standardmäßig **/usr/local/libexec**. Wenn notwendig, sollte zuerst der mysqld gestartet und die **MySQL** Grant-Tabellen initialisiert werden. Dies geschieht mittels dem Script

  ```
  mysql_install_db.
  ```

  Dies ist nur dann notwendig, wenn **MySQL** zum erstenmal installiert wird. Andernfalls liegen die **Grant-Tables** bereits vor und der Server wird mittels safe_mysql gestartet.

  ```
  scripts/mysql_install_db
  ```

  Wird dieser Schritt nicht ausgeführt, kommt es zu der Fehlermeldung:

  ```
  mysqld: Cant find file: host.frm
  ```

  Das Script erzeugt drei Tabellen (**users**, **db** und **host**) in der Datenbank **mysql.** Die voreingestellten Zugriffsberechtigungen teilen dem **MySQL**-root-Benutzer alle Berechtigungen zu und erlauben jedem Benutzer die Erzeugung und Verwendung der Datenbank **test** bzw. aller Datenbanken die mit **test_** beginnen.
- **Alternativen zum Ausführen von mysql_install_db:**
  - Vor dem Ausführen des Scripts können die Berechtigungen angepaßt werden.Dies erfolgt durch editieren des Scripts.
  - Nach der Installation des Zugriffssystems können Änderungen an den Zugriffsberechtigungen vorgenommen werden, in dem man sich mittels **mysql -u root mysql** mit der Datenbank verbindet und mittels **SQL**-Statements die Berechtigungen anpaßt.
  - Bereits vorhandene Zugriffsberechtigungen können neu initialisiert werden, indem das Script **mysql_install_db** neu gestartet wird.
- **Prüfen, ob der Server arbeitet:**

  Dies geschieht mittels **mysqladmin**:

  ```
  BINDIR/mysqladmin version
  BINDIR/mysqladmin variables
  ```

  Beispielausgabe:

  ```
  BINDIR/mysqladmin version
  mysqladmin Ver 6.3 Distrib 3.22.9-beta, for pc-linux-gnu on i686
  TCX Datakonsult AB, by Monty
  Server version 3.22.9-beta
  ```

```
Protocol version 10
Connection Localhost via UNIX socket
TCP port 3306
UNIX socket /tmp/mysql.sock
Uptime: 16 sec
```

- **Prüfen, ob der Server gestoppt werden kann:**

  BINDIR/mysqladmin -u root shutdown
- **Prüfen, ob der Server gestartet werden kann:**

  BINDIR/safe_mysql --log &
- **Ein paar Tests:**
  - ```
    BINDIR/mysqlshow

    +-----------+
    | Databases |
    +-----------+
    | mysql     |
    +-----------+
    ```
  - ```
    BINDIR/mysqlshow mysql

    Database: mysql

    +--------+
    | Tables |
    +--------+
    | db     |
    | host   |
    | user   |
    +--------+
    ```
  - ```
    BINDIR/mysql -e select host,db,user from db mysql

    +------+--------+------+
    | host | db     | user |
    +------+--------+------+
    | %    | test   |      |
    | %    | test_% |      |
    +------+--------+------+
    ```
  - Daneben steht noch eine Benchmark-Suite zur Verfügung (im Verzeichnis sql-bench) womit dei Performance von **MySQL** getestet werden kann.
  - Um alle Test durchzuführen:

    ```
    cd sql-bench
    run-all-test
    ```

    Sollte kein **sql-bench**-Verzeichnis verhanden sein, handelt es sich womöglich um eine
    ```

RPM-Binary-Distribution. (Source-RPM-Distributionen enthalten dieses Verzeichnis). Die Benchmark-Suite wird ab 3.22 separat vertrieben. Die Benchmark-RPMs sind benannt wie: **mysql-bench-VERSION-i386.rpm.**

- Ferner können alle Tests aus dem Verzeichnis **./test** ausgeführt werden. Beispiel:

```
BINDIR/mysql -wf test < ./tests/auto_increment.tst
```

erzeugt das Resultat in

```
./tests/auto_increment.res
```






25.5 Probleme beim Start von MySQL

Überprüfe das Log-File um zu sehen, ob **mysqld** korrekt gestartet wurde! Der **mysqld-Daemon** startet mit einem Verzeichnisswechsel nach **mysql-data-dir**. Danach wird **mysqld-data-dir** zu **./** (aktuelles Verzeichnis). Alle Pfade (Datenbanken, Pid-Files und Log-Files) verwenden dieses Verzeichnis als Basis-Verzeichnis. Sollten Probleme mit falsch gesetzten Pfaden auftreten, versuche mittels des Befehls

```
mysqld --help
```

die aktuell gültigen Pfade aufzulisten. Jeder Pfad kann mittels einer Startup-Einstellung in **safe_mysqld** oder **mysqld** verändert werden.

```
cd <localstatedir default /usr/local/var>
tail <Host-Name>.log
```

Um sicherzustellen, daß **MySQL** korrekt läuft, kann man folgenden Test durchführen:

```
cd /usr/local/bin
. /mysqlshow

+-----------+
| Databases |
+-----------+
| mysql     |
+-----------+

. /mysqlshow mysql

Database: mysql

+--------+
| Tables |
+--------+
| db     |
| host   |
| user   |
+--------+

./mysql -e select host,db,user from db mysql

+------+--------+------+
| host | db     | user |
+------+--------+------+
```

```
| %    | test   |      |
| %    | test_% |      |
+------+--------+------+
```

Ferner existiert eine Benchmark-Suite mit welcher die Performance von **MySQL** auf verschiedenen Plattformen getestet werden kann. In Kürze wird diese Suite auch verwendet werden können um direkte Vergleiche zwischen **MySQL** und anderen SQL-Datenbanken herzustellen.

```
cd bench
run-auto-increment-test
```

Zum Ausführen von **Auto_increment.tst** geht man wie folgt vor:

```
./mysql -vf test < ./tests/auto_increment.tst
```

Erwartete Ergebnisse werden in der Datei **./tests/auto_increment.res** dargestellt. Das **safe_mysqld**-Script wurde erstellt, um in der Lage zu sein, sowohl eine Source- als auch eine Binary-Version von mysqld starten zu können, selbst wenn sich diese in unterschiedliche Pfaden befinden. Es kann eine binäre Release von **MySQL** in jedem beliebigen Verzeichnis installieren werden , sofern **safe_mysqld** von diesem Installationsverzeichnis aus gestartet wird:

```
cd mysql_installation_directory
bin/safe_mysqld &
```

Wenn man die Startup-Einstellungen von mysqld verändern möchten, genügt es das Script **safe_mysqld** entsprechend zu verändern. Hierzu sollte man vorher das Originalscript sichern, falls bei der Anpassung etwas schief geht.






25.6 Automatischer Start/Stop von MySQL

Um **MySQL** zu Starten oder zu Stoppen verwendet man folgende Befehle:

```
scripts/mysql.server stop
scripts/mysql.server start
```

Zum automatischen Start von **MySQL** können diese Befehle in den dafür notwendigen **/etc/rc*/ Scipten** (abhängig von der verwendeten Plattform) eingebunden werden. Ferner kann das **mysql-server-Script** entsprechend angepaßt werden, um **safe_mysqld** aus einer bestimmten Umgebung heraus oder als speziellen User zu starten. Hierbei können ebenfalls weitere Startup-Optionen angegeben werden.




25.7 Globale Konfigurationsdateien (Option-Files)

- **MySQL** kann grundlegende Start-Up-Einstellungen für Server und Clients aus diversen Konfigurationsdateien lesen.
- Die Konfigurationsdateien werden wie folgt gesucht (unter Unix):

```
/etc/my.cnf Globale Konfiguration
/mysql-data-dir/my.cnf Serverkonfiguration
~./my.cnf Benutzerkonfiguration
```

  Anmerkung: das Verzeichnis **/mysql-data-dir** wird bei der Compilierung angegeben.
- **MySQL** Win32 Konfigurationsfiles

```
C:\my.cnf Globale Konfiguration
C:\mysql\data\my.cnf Serverkonfiguration
```

- Jede Einstellung aus einem vorhergehenden Konfigurationsfile wird von der gleichen Einstellung im folgenden Konfigurationsfile überschrieben. Das bedeutet, das Einstellungen aus dem Globalen Konfigurationsfile von den Einstellungen des Benutzerkonfigurationsfiles überschrieben werden. Kommandozeilen-Optionen überschreiben alle Konfigurationsfile-Einstellungen.
- Konfigurationsfiles werden von den folgenden Programmen unterstützt:

```
mysql
mysqladmin
mysqld
mysqldump
mysqlimport
isamchk
pack_isam
```

  Um eine Übersicht zu erhalten, welche Optionen von den jeweiligen Programmen unterstützt werden, ist das entsprechende Programm mit dem Parameter **--help aufzurufen**.
- Ein Konfigurationsfile ist wie folgt aufgebaut:
  - **#:** Kommentare beginnen mit #.
  - Alle Leerzeilen werden ignoriert. Ein Beispiel für ein Konfigigurationsscript ist in der Datei **my-example.cnf** zu finden, welches im **bin-** bzw. **scripts**-Verzeichnis abgelegt ist. Folgende vordefinierten Gruppen sind verfügbar: Client Konfiguration wird für alle **MySQL**-Clients verwendet, aber **nicht** für **mysqld**.
  - Im folgenden einige Beispiele:

- **Globales Options-File:**

  [client]

  port=3306

  socket=/tmp/mysql.sock

  [mysqld]

  port=3306

  socket=/tmp/mysql.sock

  set-variable = key_buffer=16M

  set-variable = max_allowed_packet=1M

  [mysqldump]

  quick

Sollen keine Konfigurationsdateien verwendet werden, so ist ist das jeweilige Programm mit dem Parameter **--no-defaults** zu starten. Dieser Parameter **muß!!** der erste Parameter sein, der an das jeweilige Programm übergeben wird.






26. Zugriffsberechtigungen (Privilege-System)

Direkt nach der Installation sollte man die Zugriffsrechte auf die MySQL Datenbank einschränken. Hierzu sollte das Einsteigertutorial MySQL für LINUX allen Einsteigern eine solides Verständnis der Materie vermitteln. Siehe Kapitel Einsteiger - Tutorial LINUX MySQL Server. Wer nun tiefer in die Vergabe der Rechte einsteigen möchte, für den sind die folgenden Kapitel von Interesse:







26.1 Einleitung

MySQL verwendet ein leistungsstarkes, nicht standardisiertes Sicherheits/Zugriffs-System, welches im Folgenden näher dargestellt werden soll.

Vorab sei erwähnt, daß nach jeder Änderung der Zugriffsrechte ein Reload des DB-Servers durchgeführt werden muß, damit die Änderungen wirksam werden. Dies wird mittels: **mysqladmin reload** erledigt.







26.2 Welche Aufgaben werden vom Berechtigungssystem erledigt?

Die grundlegende Funktion besteht darin, einem Benutzer auf einem Host Berechtigungen zur Ausführung von **SELECT**, **INSERT**, **UPDATE** und **DELETE** Befehlen zu erteilen (oder auch nicht). Ferner kann die Verwendung eines **Anonymen Benutzer** sowie **MySQL**-Funktionen wie **LOAD DATA INFILE** geregelt werden. Weiterhin werden mittels des Berechtigungssystems administrative Rechte an dem DB-Server selbst, wie z.B. **Reload**, vergeben.

- **Anmerkung:** Man sollte im Folgenden beachten, daß die **MySQL** Usernamen und Passwörter nichts mit den UNIX-Usernames und Passwörtern zu tun haben!







26.3 Wie funktioniert das Sicherheitssystem?

MySQL unterscheidet nicht zwischen einzelnen Benutzern sondern zwischen der Kombination aus Benutzer und Host. Diese Kombination ergibt eine eindeutige ID. Das Sicherheitssystem regelt hierbei genauestens welcher Benutzer von welchem Host welche Rechte auf welcher Datenbank besitzt.

Diese Einstellungen werden in drei Tabellen abgelegt:

```
user
host
db
```

Die jeweiligen Tabellen sind wie folgt aufgebaut:







26.4 user-Tabelle

Beinhaltet alle Host/User Kombinationen, welche den **MySQL**-Server connecten dürfen. Alle Berechtigungen die ein Benutzer in dieser Tabelle enthält gelten für alle Datenbanken, sofern keine erweiterten Berechtigungen für den jeweiligen Benutzer in der Tabelle **db** definiert wurden. Man kann diese Berechtigungen auch als grundlegende Einstellungen ansehen und ein datenbankabhängiges Fein-Tunig in der Tabelle **db** festlegen.

Der Tabellenaufbau:

```
Field Type Key Default
-----------------------------------------------------------------
Host char(60) PRI
User char(16) PRI
Password char(16) -
Select_priv enum(N,Y) - N
Insert_priv enum(N,Y) - N
Update_priv enum(N,Y) - N
Delete_priv enum(N,Y) - N
Create_priv enum(N,Y) - N
Drop_priv enum(N,Y) - N
Reload_priv enum(N,Y) - N
Shutdown_priv enum(N,Y) - N
Process_priv enum(N,Y) - N
File_priv enum(N,Y) - N
```







26.5 db-Tabelle

In dieser Tabelle wird definiert welche Datenbank der jeweilige Host/Benutzer mit welchen Berechtigungen verwenden darf. Sie stellt wie bereits erwähnt das Fein-Tunig da.

Der Tabellenaufbau:

```
Field Type Key Default
-----------------------------------------------------------------
Host char(60) PRI
Db char(64) PRI
User char(16) PRI
Select_priv enum(N,Y) - N
Insert_priv enum(N,Y) - N
Update_priv enum(N,Y) - N
Delete_priv enum(N,Y) - N
Create_priv enum(N,Y) - N
Drop_priv enum(N,Y) - N
```







26.6 host-Tabelle

Die **host**-Tabelle ist in großen Netzwerken als **Nachschlage**-Tabelle für leere Host-Einträge in der **db**-Tabelle sinnvoll. Möchte man, daß ein Benutzer von jedem Host in dem Netzwerk auf den DB-Server zugreifen kann, sollte man den Host-Eintrag in der **db**-Tabelle auslassen und alle Host des Netzwerkes in der **host**-Tabelle eintragen.

Der Tabellenaufbau:

```
Field Type Key Default
-------------------------------------------------------------
Host char(60) PRI
Db char(64) PRI
Select_priv enum(N,Y) - N
Insert_priv enum(N,Y) - N
Update_priv enum(N,Y) - N
Delete_priv enum(N,Y) - N
Create_priv enum(N,Y) - N
Drop_priv enum(N,Y) - N
```







26.7 Definitionen

Die **HOST** und **DB** Spalten können Strings mit Wildcards **%** und **_** beinhalten. Wird für diese Spalten kein Wert eingetragen entspricht dies dem Wert **%**.

Ein HOST kann sein:

```
localhost
ein Hostname
eine IP-Nummer
ein String mit Wildcards
```

Ein leerer HOST-Eintrag in der **db**-Tabelle bedeutet -kein Host- aus der **host**-Tabelle. Ein leerer HOST-Eintag in der **host**- oder **user**-Tabelle bedeutet -kein Host-.

Die Spalte **DB** beinhaltet den Namen einer Datenbank oder einer **SQL Regexp**.

Ein leerer Benutzereintrag bedeutet -kein Benutzer-. In dieser Spalte können keine Wildcards verwendet werden.

Die Berechtigungen der **user**-Tabelle werden ge-**OR**-d mit den Berechtigungen aus der **db**-Tabelle. Dies Bedeutet, daß ein **Superuser** nur in der Tabelle **user** mit allen Berechtigungen festgelegt auf **Y** eingetragen werden muß.

Wenn man sich nun den Aufbau der Tabellen näher betrachtet, wird man feststellen, daß die **user**-Tabelle zusätzlich zu den Zugriffsberechtigungen auf die jeweilige Datenbank auch administrative Berechtigungen regelt. Dadurch sollte klar sein, daß diese Tabelle die grundlegenden Berechtigungen regelt.







26.8 Festlegen von Berechtigungen

Im folgenden Beispiel soll der Benutzer **custom** mit dem Paßwort **stupid** angelegt werden.Der Benutzer soll die Möglichkeit haben sich von den Hosts **localhost**, **server.domain** und **whitehouse.gov** anzumelden. Die Datenbank **bankaccount** möchte er nur von dem Host **localhost** und die Datenbank **customer** von allen Hosts ansprechen können.

```
> mysql -u root mysql.

mysql> insert into user (host,user,password)
values(localhost,custom,password(stupid));

mysql> insert into user (host,user,password)
values(server.domain,custom,password(stupid));

mysql> insert into user (host,user,password)
values(whitehouse.gov,custom,password(stupid));

mysql> insert into db
(host,db,user,Select_priv,Insert_priv,Update_priv,Delete_priv,
Create_priv,Drop_priv)
values
(localhost,bankaccount,custom,Y,Y,Y,Y,Y,Y);

mysql> insert into db
(host,db,user,Select_priv,Insert_priv,Update_priv,Delete_priv,
Create_priv,Drop_priv)
values
(%,customers,custom,Y,Y,Y,Y,Y,Y);

mysql> quit

> mysqladmin reload
```







26.9 Weitere Beispiele:

Dieses Beispiel zeigt, wie man auf die Datenbank **mysql** direkt zugreift.

```
shell> mysql -u root mysql

mysql> insert into user values
(%,monty,password(something),Y,Y,Y,Y,Y,Y,Y,Y,Y,Y)
;

mysql> insert into user (host,user,password) values(localhost,dummy,)
;

mysql> insert into user values
(%,admin,,N,N,N,N,N,N,Y,N,Y,Y)

mysql> quit
```

Dieses Beispiel legt drei neue Benutzer an:

- **monty** als Superuser (alle Berechtigungen); muß zum Anmelden das Paßwort **something** verwenden.
- **admin** Hat nur Berechtigung zur Ausführung von:

  ```
  mysqladmin reload
  mysqladmin refresh
  und
  mysqladmin processlist
  ```

  Weitere Berechtigungen für diesen Benutzer beim Zugriff auf Datenbanken müssen in der **db**-Tabelle definiert werden.
- **dummy** Diesem Benutzer müssen auch mittels der **db**-Tabelle weitere Berechtigungen zum Zugriff auf entsprechende Datenbanken zugeteilt werden.







26.10 Default-Berechtigungen

Die Default-Berechtigungen, festgelegt in **scripts/mysql_install-db**, geben dem Benutzer root alle Berechtigungen. Jeder Benutzer hat alle Berechtigungen auf die Datenbanken mit dem Namen **test** oder welche mit **test_** beginnen. Ein normaler Benutzer hat keine Berechtigung.

```
mysqladmin shutdown
oder
mysql processlist
auszuführen.
```







27. Zugriffsprobleme

Gerade für Einsteiger sind einige Probleme ein unüberwindbares Hindernis und kosten viel Zeit und Recherche im Internet. Standardmäßig sind daher alle MySQL Installationen möglichst offen und stellen auch ein Sicherheitsproblem dar. Die Inbetriebnahme einer solchen Standardinstallation ist im Tutorial Einsteiger Tutorial MySQL unter LINUX ausführlich für Einsteiger erklärt !

Die Fragen und Probleme im folgenden Kapitel rekrutieren sich aus den gesammelten Anfragen bei den Entwicklern von MySQL.






27.1 Checkliste bei Zugriffsproblemen

Wieso es beim Zugriff auf **MySQL** zu **Access denied**-Meldungen kommen kann, hat viele mögliche Ursachen. Die folgende Checkliste sollte deshalb gründlich durchgearbeitet werden.

- Wurden die **MySQL** Grant-Tabellen mittels des Scipts: **mysql_install_db** erzeugt ? Um dies zu testen, sollte der Befehl **mysql -u root test** ausgeführt werden, welcher keine Fehlermeldung erzeugen sollte. Ferner kann geprüft werden, ob eine Datei mit dem Namen **user.ISD** im Datenbankverzeichnis (**Installationsverzeichnis/var/mysql/user.ISD**)vorhanden ist.
- Nach einer Erstinstallation sollte mittels **mysql -u root mysql** eine Verbindung auf die Grant-Tabellen erfolgen und alle Rechte gesetzt werden.
- Nach jeder Änderung in der Zugriffsberechtigung ist ein **mysqladmin reload** erforderlich, damit die Änderungen auch greifen. Zum testen kann der **mysqld**-Daemon mit der Option **--without-grant-tables** gestartet werden. Dann können die Grant-Tabellen angepaßt und mittels **mysqlaccess** überprüft werden. Mittels **mysqladmin reload** wird der Daemon mit den geänderten Grant-Tabellen neu gestartet.
- Auch bei Zugriffsproblemen von PERL, Python oder ODBC aus sollten die Tests mittels **mysql -u User Database** oder **mysql -u User -pPassword Database** durchgeführt werden. Zwischen dem Schalter **-p** und dem eigentlichen Paßwort ist kein Leerzeichen erlaubt. Die Syntax **--password=Password**kann statt dessen verwendet werden.
- Kommt es zu der Meldung **Access denied** beim Zugriff mittels **mysql -u user database** liegt dies an einer nicht korrekten Einstellung in der **user**-Tabelle. Zur Überprüfung sollte man sich mittels **mysql -u root mysql** verbinden und die Tabellen durch das Kommando **select * from user** anzeigen lassen. In der Tabelle muß auf alle Fälle ein Eintrag für **User** und **Hostname** enthalten sein, welcher mit dem aktuellen Benutzer und Host übereinstimmt. Werden Client und Server auf dem gleichen Host ausgeführt, und es wurde nicht die **mysql**- Option **--host** sowie das MIT-pthread-Packet verwendet, ist **localhost** ein Synonym für den Hostnamen.
- Die **Access denied**-Meldung zeigt zusätzlich an, mit welchem Benutzernamen, von welchem Host aus man einen Connect aufbauen wollte und ob man ein Paßwort verwendet oder nicht. In der **user**-Tabelle sollte ebenfalls ein Eintrag enthalten sein, der mit diesen Angaben (User und Hostname) genau übereinstimmt.
- Kommt es zu der Meldung **Host... is not allowed to connect to this MySQL server** beim Verbindungsaufbau von einem anderen Host aus, ist in der **user** Tabelle der entsprechende Eintrag, welcher diesen Host betrifft nicht vorhanden. Der fehlende Eintrag sollte auf bekannte Weise erstellt werden. Verwendet man nicht **MySQL** 3.22 und man kennt nicht die IP bzw. den Hostnamen, von welchem man sich connecten will, sollte **&** als Hostname in der Tabelle **user** eingetragen und der mysql-Daemon mit der Option **-log** neu gestartet werden. Nach einem erneuten Verbindungsversuch findet man in dem **MySQL**-Logfile die genaue Bezeichnung des Hostnamens.
- Wenn eine Verbindung mittels **mysql -u root test** funktioniert aber **mysql -h hostname -u root test** die Meldung **Access denied...** verursacht, liegt dies daran, daß der Hostname in der **user**-Tabelle nicht korrekt ist. Dies kann dann passieren, wenn der Hostname z.B. als **tcx**

definiert wurde der DNS aber den Namen als **tcx.subnet.se** auflöst. Zum testen dieser Situation sollte man statt des Hostnamens die IP-Adresse eintragen. Alternativ kann auch der Hostname mittels **tcx%** eingetragen werden, was aber sehr unsicher ist.

- Wenn man nicht feststellen kann, weshalb weiterhin die Meldung **Access denied...** erzeugt wird, sollte man alle Hostnamen-Einträge die einen Wildcard (**%**< oder **_**) beinhalten aus der **user**- Tabelle entfernen. Oft geht man davon aus, daß man mittels des Eintrags **Host=%** und **user=User** die Möglichkeit hat, sich von dem lokalen Host (localhost) aus zu connecten. Der Grund, warum dies nicht funktioniert ist, daß der Eintrag host=**localhost** und user= vor dem Eintrag host=**%** und user=**user** bevorzugt wird! Um dies zu beheben sollte ein Eintrag host=**localhost** und user= **user** erstellt oder der Eintrag user= entfernt werden.
- Werden die MIT-pthreads verwendet, wird **localhost** nie verwendet! Alle Verbindungen zu dem mysqld-Daemon erfolgen mittels **TCP/IP**, weshalb der richtige Hostname in der **user**-Tabelle vorhanden sein muß, selbst dann, wenn der Client auf dem selben Host ausgeführt wird wie der Server.
- Bei der Fehlermeldung **Access to database denied** liegt ein Problem in der **db**-Tabelle vor. Beinhaltet der entsprechende Eintrag in der **db**-Tabelle einen leeren Hostname sollte ebenfalls die **host**-Tabelle überprüft werden.
- Funktioniert der Befehl **mysql -u user database** auf der Server-Maschine aber nicht **mysql -h host -u user database** auf einer anderen Client-Maschine, dann fehlt der entsprechende Eintrag in der **user** oder **db**-Tabelle für diese Client-Maschine.
- Alle Paßwörter müssen mit der **PASSWORD**-Funktion eingegeben werden.
- Wenn **mysql -u user test** funktioniert, aber **mysql -u user andere_database** nicht, fehlt der entsprechende Eintrag für **andere_database** in der **db-Tabelle**.
- Kommt es bei den SQL-Befehl **SELECT...INTO OUTFILE** oder **LOAD DATA** zu der Meldung: **Access to database denied** wurden vermutlich die **File_priv** Berechtigungen in der **user**-Tabelle nicht gesetzt.
- Sollte die Fehlersuche dennoch negativ verlaufen, ist der **mysqld-Daemon** mit der Option: **--debug=d,general,query** zu starten. Dadurch werden Informationen zu dem Host und User, welcher eine Verbindung aufbauen will, sowie über jedes ausgeführte Kommando ausgegeben. Ist der Fehler dennoch nicht zu finden und man möchte eine entsprechende Anfrage an die Mailingliste richten, sollte ein Auszug der Grant-Tabellen beigefügt werden.
- Bei der Meldung **Cant connect to local mySQL server** oder **Cant connect to MySQL server on hostname** kann davon ausgegangen werden, daß der mysqld-Daemon nicht läuft oder daß man versucht auf einen falschen Port oder Socket zu connecten. Hierzu sollte man prüfen, ob der Socket vorhanden ist (normalerweise **/tmp/mysql.socket**) oder eine Verbindung mittels **telnet hostname 3306** aufbauen. Durch **mysqladmin version** können ebenfalls weitere Informationen ausgegeben werden.
- Verwendet man einen Client ohne Angabe von Parametern sollte man prüfen ob ggf. noch falsche Paßwörter in den Konfigurations-Dateien vorhanden sind.






28. MySQL-Sprachreferenz mit vielen Beispielen

Diese Sprachreferenz lehnt sich eng am originalen Handbuch MySQL 3.23 im Internet an. Im Grunde wurden fast alle Kapitel 1:1 übernommen und ggf. ergänzt. Es beschreibt die Version 3.23+. Falls also einige wenige Befehle nicht funktionieren, dann könnte es an Ihrer etwas veralteten MySQL Version liegen. Schauen Sie hierzu im Kapitel Neuigkeiten bei MySQL 3.23 nach.

MySQL ist in vielen Punkten ebenso leistungsfähig, in einigen mathematischen Funktionen sogar leistungsfähiger, als kommerzielle Datenbanken. Man sollte sich einmal diese Sprachreferenz genau durchlesen, um einen Überblick zu haben, was alles möglich ist.







28.1 CREATE TABLE

```
CREATE TABLE

CREATE TABLE tbl_name (create_definition,...)

create_definition:
  col_name type [NOT NULL | NULL] [DEFAULT default_value] [AUTO_INCREMENT]
            [PRIMARY KEY] [reference_definition]
  oder  PRIMARY KEY (index_col_name,...)
  oder  KEY [index_name] KEY(index_col_name,...)
  oder  INDEX [index_name] (index_col_name,...)
  oder  UNIQUE [index_name] (index_col_name,...)
  oder  [CONSTRAINT symbol] FOREIGN KEY index_name (index_col_name,...)
            [reference_definition]
  oder  CHECK (expr)

Typ:
        TINYINT[(length)] [UNSIGNED] [ZEROFILL]
  oder  SMALLINT[(length)] [UNSIGNED] [ZEROFILL]
  oder  MEDIUMINT[(length)] [UNSIGNED] [ZEROFILL]
  oder  INT[(length)] [UNSIGNED] [ZEROFILL]
  oder  INTEGER[(length)] [UNSIGNED] [ZEROFILL]
  oder  BIGINT[(length)] [UNSIGNED] [ZEROFILL]
  oder  REAL[(length,decimals)] [UNSIGNED] [ZEROFILL]
  oder  DOUBLE[(length,decimals)] [UNSIGNED] [ZEROFILL]
  oder  FLOAT[(length,decimals)] [UNSIGNED] [ZEROFILL]
  oder  DECIMAL(length,decimals) [UNSIGNED] [ZEROFILL]
  oder  NUMERIC(length,decimals) [UNSIGNED] [ZEROFILL]
  oder  CHAR(length) [BINARY]
  oder  VARCHAR(length) [BINARY]
  oder  DATE
  oder  TIME
  oder  TIMESTAMP
  oder  DATETIME
  oder  TINYBLOB
  oder  BLOB
  oder  MEDIUMBLOB
  oder  LONGBLOB
  oder  TINYTEXT
  oder  TEXT
  oder  MEDIUMTEXT
  oder  LONGTEXT
  oder  ENUM(value1,value2,value3...)
  oder  SET(value1,value2,value3...)

index_col_name:
        col_name [(length)]

reference_definition:
        REFERENCES tbl_name [(index_col_name,...)]
                   [MATCH FULL | MATCH PARTIAL]
                   [ON DELETE reference_option]
                   [ON UPDATE reference_option]
```

```
reference_option:
        RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT

table_options:
        type = [ISAM | MYISAM | HEAP]
oder      auto_increment = #
oder      avg_row_length = #
oder      checksum = [0 | 1]
oder      comment = "string"
oder      max_rows = #
oder      min_rows = #
oder      pack_keys = [0 | 1]
oder      password= "string"

select_statement:
        [ | IGNORE | REPLACE] SELECT ...  (Hier kann irgendein SELECT Statement
eingefügt werden)
```

CREATE TABLE erstellt eine neue Tabelle in der aktuellen Datenbank (Datenbank kann mittels **USE db_name** ausgewählt werden).

Jede Tabelle wird durch drei Dateien im Datenbankverzeichnis dargestellt:

```
tbl_name.frm - Tabellendefinition
tbl_name.ISD - Datenfile
tbl_name.ISM - Indexfile
```

Anmerkung: ab **MySQL** 3.22 kann der Tabellenname mittels **db_name.tbl_name** angegeben werden.

Was hat es mit den drei Dateien auf sich ? In der Tabellendefinition befindet sich die Struktur der Datenbank, die Angaben über die Größen der Felder u.s.w. Im Datenfile befinden sich alle Daten, die dort so abgelegt sind, daß die schnell wiedergefunden werden können. Im Indexfile ist die Datenbank als HASH-Tabelle abgelegt, welches die Suche im Datenfile drastisch beschleunigt.

Zur Syntax von CREATE

Der CREATE Befehl ist besonders wichtig, da ohne eine solide Datenbankstruktur nichts korrekt läuft. Man sollte sich gut überlegen, welche Typen und Attribute man verwendet, sowie man die Datenbank am geschicktesten anlegt. Spätere Korrekturen sind schlecht möglich. Bevor man nun einfach eine Datenbank erzeugt, sollte man das Kapitel Datenbank Grundlagen und das Kapitel zur Normalisierug von Datenbanken

- Wird eine Spalte ohne **NULL** oder **NOT NULL** definiert, wird diese automatisch als NULL definiert.
- Eine Spalte vom Typ INTEGER kann zusätzlich das Attribut **AUTO_INCREMENT** erhalten. Wodurch automatisch nach dem Einfügen eines neuen Datensatzes dieses Feld einen um 1 höheren Wert erhält als der höchste in dieser Spalte vorhandene Wert. Wird die Spalte wieder gelöscht, so steht dieser Feldwert bei dem nächsten Datensatz wieder zur Verfügung. Anmerkung: Pro Tabelle kann immer nur 1 Feld das **AUTO_INCREMENT** Attribut erhalten.
- NULL-Werte werden bei Feldern vom Typ **TIMESTAMP** ignoriert. In diesen Feldern kann nicht der Wert NULL abgelegt werden. In diesem Fall wird immer das aktuelle Datum abgelegt. Hierbei ist zu beachten, daß der Wert **0** nicht dem Wert **NULL** entspricht. Der Wert 0 ist wiederum ein zulässiger Wert, welcher in solch einem Feld abgelegt werden kann.
- Wird für eine Spalte kein **DEFAULT** vorgegeben und wird diese als **NOT NULL** definiert, wird der Wert **NULL** als **DEFAULT** verwendet.
- Wird eine Spalte ohne **DEFAULT**-Wert angelegt und als **NOT NULL** definiert, legt **MySQL**, abhängig vom Datentyp einen Defaultwert automatisch fest.
    - Bei nummerischen Datentypen ist dies 0.
    - Bei DATE und TIME Typen ist dies der **Startwert** dieses Typs.
    - Bei String-Typen ist dies ein leerer String.
- **KEY** ist ein Synonym für **INDEX**
- Ein **UNIQUE KEY** kann nur eindeutige Werte besitzen. Wird mehrmals versucht den gleichen Wert zu verwenden, kommt es zu einer Fehlermeldung

- Ein **PRIMARY KEY** stellt einen eindeutigen Key dar. Jede Tabelle kann maximal einen **PRIMARY KEY** besitzen. **MySQL** legt den ersten **UNIQUE KEY** als **PRIMARY KEY** fest, sofern kein **PRIMARY KEY** definiert wurde
- Ein **PRIMARY KEY** kann aus mehreren Spalten zusammengesetzt sein (bis zu 16 bzw. 32 (3.23.x). Allerdings darf ein mehrspaltiger Index keine **PRIMARY KEY** Spalte enthalten. Dies führt zu einer Fehlermeldung.
- Wird beim Definieren eines **INDEX** kein Name für diesen **INDEX** angegeben, so wird der Name des ersten Feldes, daß im Index enthalten ist, mit einem Suffix z.B. _2, _3, ... verwendet.
- Spalten, welche indiziert oder Teil eines **INDEX** sind, können keine **NULL** Werte enthalten. Diese Spalten müssen als **NOT NULL** deklariert werden.
- Mit der **col_name**(länge)-Syntaxe kann der Teil eines **CHAR** oder **VARCHAR** feldes festgelegt werden, der in den Index einfließt.
- **TEXT** und **BLOB** Felder können nicht iniziert sein.
- Die **FOREIGN KEY**, **CHECK** und **REFERENCES** Klausen haben zur Zeit keinerlei Auswirkung. Diese Syntax wird nur zur Verfügung gestellt um kompatibel zu anderen SQL-Servern zu sein.
- In manchen Fällen ändert **MySQL** automatisch die Felddefinitionen, welche in der **CREATE**-Anweisung enhalten sind um:
    - **VARCHAR** Felder mit einer Länge kleiner 4 werden in **CHAR** umgewandelt
    - **TIMESTAMP** Felder müssen eine Länge von mindesten 2 und maximal 14 haben. Wird ein Wert auserhalb dieser Range angegeben, wird er auf 14 gesetzt.

Die Optionen table_options und einige SELECT Optionen gibt es erst ab der MySQL Version 3.23. Hier wurden folgende Typen hinzugefügt:

- **ISAM** Der originale Tabellen Handler (default Einstellung)
- **MyISAM** Ist die neue, für SQL optimierte MyISAM Datenbankstruktur. Diese ist nicht zum Vorgänger kompatibel. Falls Sie also updaten möchten, müssen Sie alle Daten zwischen zwei MySQL Datenbanken kopieren. (Setzen Sie einfach hierzu eine neue MySQL Datenbank auf einem anderen Port auf)
- **HEAP** Ist die Datenstruktur für eine Tabelle, die nur im RAM abgelegt wird. Entsprechend sind die Zugriffe darauf ca. 5-30x schneller, als auf die normale Datenbank. Dies kann VIEWS vollständig ersetzen.

Hier gibt es noch ein paar Optionen, die nur in Tabellen mit der neuen, MyISAM Struktur funktionieren:

- **checksum** ist ganz nützlich, wenn man eine Korruption in der Datenstruktur feststellen können muß. (Wert auf 1 setzen)
- **max_rows** ist der Wert der maximalen Spalten in einer Tabelle und wichtig für ISP´s, die die Auswirkungen von unsinnigen CREATE Statements klein halten möchten.
- **min_rows** ist der entsprechende Minimalwert
- **pack_keys** sollte auf 1 gesetzt werden, damit der INDEX kleiner wird. Bei Updates ist MySQL etwas langsamer, beim Lesen viel schneller.
- **password** verschlüsselt die .frm Datei mit einem Passwort

Beispiele zu CREATE und ALTER

Hier sind einige Beispiele die zeigen, wie man mit CREATE und ALTER TABLE umgeht:

```
mysql> CREATE TABLE test (a int not null auto_increment, primary key (a), key(b))
     TYPE=HEAP SELECT b,c from test2;
```

Dieses Beispiel legt eine temporäre HEAP Tabelle mit 3 Spalten an. Diese HEAP Tabelle wird vollständig im RAM abgelegt. Eventuell werden Teile auch im SWAP - File ausgelagert (erst ab Version 3.23). Die Tabelle kann nachher wieder gelöscht werden. Falls ein Fehler auftritt, dann wird diese Tabelle automatisch gelöscht.

```
mysql> CREATE TABLE t1 (a INTEGER,b CHAR(10));
```

Um Tabelle von t1 nach t2 umzubenennen:

```
mysql> ALTER TABLE t1 RENAME t2;
```

Um eine Spalte von INTEGER nach TINYINT NOT NULL zu konvertieren, und gleichzeitig Spalte b von CHAR(10) nach CHAR(20) unter Umbenennung von b nach c zu verändern:

```
mysql> ALTER TABLE t2 MODIFY a TINYINT NOT NULL, CHANGE b c CHAR(20);
```

Einfügen einer TIMESTAMP Spalte mit Namen d:

```
mysql> ALTER TABLE t2 ADD d TIMESTAMP;
```

Um einen Index an Spalte d anzufügen, und diese als primary key zu bestimmen:

```
mysql> ALTER TABLE t2 ADD INDEX (d), ADD PRIMARY KEY (a);
```

Spalte c Löschen:

```
mysql> ALTER TABLE t2 DROP COLUMN c;
```

Hinzufügen einer AUTO_INCREMENT integer Spalte namens c:

```
mysql> ALTER TABLE t2 ADD c INT UNSIGNED NOT NULL AUTO_INCREMENT,
           ADD INDEX (c);
```

Man beachte, daß wir c indiziert haben, weil AUTO_INCREMENT Spalten zwangsläufig indiziert werden müssen. Wir erklären c als NOT NULL, weil indizierte Spalten nicht NULL sein dürfen.








28.2 OPTIMIZE TABLE SYNTAX

Die Syntax lautet:

```
OPTIMIZE TABLE tbl_name
```

OPTIMIZE TABLE sollte verwendet werden, wenn in einer Tabelle viele Änderungen oder größere Löschvorgänge stattgefunden haben. Es wird oft viel Speicher wieder freigegeben. Hierzu wird intern eine Kopie der Tabelle angelegt, also Achtung beim Speicherverbrauch.







28.3 ALTER Table

```
ALTER [IGNORE] TABLE tbl_name alter_spec [, alter_spec ...]

alter_specification:
          ADD [COLUMN] create_definition [FIRST | AFTER column_name ]
  oder    ADD INDEX [index_name] (index_col_name,...)
  oder    ADD PRIMARY KEY (index_col_name,...)
  oder    ADD UNIQUE [index_name] (index_col_name,...)
  oder    ALTER [COLUMN] col_name {SET DEFAULT literal | DROP DEFAULT}
  oder    CHANGE [COLUMN] old_col_name create_definition
  oder    MODIFY [COLUMN] create_definition
  oder    DROP [COLUMN] col_name
  oder    DROP PRIMARY KEY
  oder    DROP INDEX key_name
  oder    RENAME [AS] new_tbl_name
  oder    table_option
```

ALTER erlaubt es, die Struktur einer Tabelle vollständig zu verändern. Intern legt MySQL eine temporäre Kopie dieser Tabelle an, verändert die Struktur und kopiert die Daten wieder zurück. Bei großen Datenbanken kann dies erhebliche Zeit in Anspruch nehmen.

```
mysql> CREATE TABLE t1 (a INTEGER,b CHAR(10));
```

Um Tabelle von t1 nach t2 umzubenennen:

```
mysql> ALTER TABLE t1 RENAME t2;
```

Um z.B. eine INTEGER Spalte von a nach b umzubenennen:

```
mysql> ALTER TABLE t1 CHANGE a b INTEGER;
```

Um einen Spaltentyp zu verändern, ohne den Namen zu verändern:

```
mysql> ALTER TABLE t1 CHANGE b b INTEGER;
```

Ab Version 3.22.16a funktioniert auch folgende Syntax:

```
mysql> ALTER TABLE t1 MODIFY b INTEGER;
```

Um eine Spalte von INTEGER nach TINYINT NOT NULL zu konvertieren, und gleichzeitig Spalte b von CHAR(10) nach CHAR(20) unter Umbenennung von b nach c zu verändern:

```
mysql> ALTER TABLE t2 MODIFY a TINYINT NOT NULL, CHANGE b c CHAR(20);
```

Einfügen einer TIMESTAMP Spalte mit Namen d:

```
mysql> ALTER TABLE t2 ADD d TIMESTAMP;
```

Um einen Index an Spalte d anzufügen, und diese als primary key zu bestimmen:

```
mysql> ALTER TABLE t2 ADD INDEX (d), ADD PRIMARY KEY (a);
```

Spalte c Löschen:

```
mysql> ALTER TABLE t2 DROP COLUMN c;
```

Hinzufügen einer AUTO_INCREMENT integer Spalte namens c:

```
mysql> ALTER TABLE t2 ADD c INT UNSIGNED NOT NULL AUTO_INCREMENT,
           ADD INDEX (c);
```

Man beachte, daß wir c indiziert haben, weil AUTO_INCREMENT Spalten zwangsläufig indiziert werden müssen. Wir erklären c als NOT NULL, weil indizierte Spalten nicht NULL sein dürfen.

Einige Dinge sollten Sie bei ALTER unbedingt beachten:

- Um ALTER TABLE verwenden zu können, ist es erforderlich, daß Sie die Rechte **select, insert, delete, update, create** und **drop** an der Tabelle besitzen. Das ist angesichts der internen Abläufe in MySQL beim ALTER Statement auch verständlich.
- IGNORE ist eine MySQL Erweiterung gegenüber ANSI SQL 92. Sie kontrolliert, die ALTER TABLE arbeitet, wenn es Duplikate bei UNIXQUE Keys einer neuen Tabelle gibt. Wenn IGNORE nicht angegeben ist, wird die Copie abgebrochen und ds Statement ignoriert. Wenn aber IGNORE angegeben wurde, dann wird nur die erste Spalte verwendet, wenn ein UNIQUE Key angegeben ist, alle anderen Spalten werden gelöcht.
- Es dürfen mehrere ADD, ALTER, DROP und CHANGE Klauseln in einer einzigen ALTER TABLE Statement verwendet werden.
- CHANGE col_name, DROP col_name und DROP INDEX sind erweiterungen von MySQL gegenüber ANSI SQL 92.
- MODIFY ist eine ORACLE Erweiterung zu ALTER TABLE.
- Das Wort COLUMN kann auch weggelassen werden.
- Wenn ALTER TABLE tbl_name RENAME AS neuer_name ohne irgendwelche Optionen verwendet wird, werden die Dateinamen auf der Festplatte einfach umbenannt. Es werden keine temporären Tabellen angelegt, die später eventuell gelöscht werden müßten.
- Falls durch CHANGE oder MODIFY eine Spalte verkürtzt wird, auf welchem ein INDEX liegt, dann wird die Länge nur auf die Länge des INDEX verkürzt, damit der INDEX noch arbeitsfähig ist. Andernfalls müssen sie den INDEX löschen, die Spalte verändern, und den INDEX neu erstellen.
- Mit MySLQ ab Version 3.22 darf nun der Ausdruck FIRST oder ADD... AFTER col_name verwendet werden, um eine Spalte nach einer bestimmten Position einzufügen. Standardmäßig wird am Ende stets eine Spalte angefügt.

- DROP INDEX entfernt einen INDEX auf einer Spalte
- DROP PRIMARY KEY löscht den primären INDEX. Falls dieser nicht existiert, so wird diejenige Spalte verworfen, die den ersten UNIQUE Key ist.
- Mit der C API Funktion mysql_info() kann man herausfinden, wieviele Einträge kopiert wurden, und bei der Verwendung von IGNORE wird angegeben, wieviele Einträge gelöscht wurden.
- Die Ausdrücke FOREIGN KEY, CHECK und REFERENCES haben keine Funktion.






28.4 DROP Table

```
DROP TABLE [IF EXISTS] tbl_name [, tbl_name,...]
```

DROP TABLE entfernt eine oder mehrere Tabellen sowie deren Definitionen, also Vorsicht ! In MySQL 3.22 oder später verhindern die Schlüsselworte IF EXISTS einen ERROR von TABELLEN, die nicht existieren.







28.5 Typenveränderungen durch den MySQL Optimizer

Falls sie sich eventuell wundern, daß ein von Ihnen angelegter Spaltentyp verändert wurde, dann hat das den Grund, daß der Optimizer von MySQL die Struktur zwecks höherer Performance geändert hat:

- VARCHAR mit einer Länge von weniger als 4 Buchstaben werden automatisch in ein Array von CHAR() umgewandelt. Alle Spalten, die als Spalten mit variabler Länge definiert wurden (VARCHAR, TEXT, BLOB), bewirken, daß auch die ganze Zeile (Reihe) als vaiable Länge gespeichert wird. (Siehe auch rowformat)
- Spalten, bei denen alle Spalten länger als 3 Buchstaben sind, werden automatisch nach VARCHAR konvertiert.
- Ungerade Längen von Spaltentypen werden auf Längen mit geraden Zahlen gerundet.
- TIMESTAMP Größen müssten zwischen 2 und 14 Bytes betragen. Größere Werte werden auf 14 zurückgesetzt.
- Wenn TIMESTAMP Spalten auf das Attribut NULL gesetzt werden, dann wird stets nur die aktuelle Zeit darin eingetragen.
- In Verbindung mit ODBC Treiber und ACCESS, z.B. wandelt MySQL automatisch bestimmte Typen (TEXT, MEMO) in eigene Formate um. Das Format ist in Abhängigkeit von der ACCESS Version und dem MyODBC Treiber stets verschieden (ACCESS hat viele verschiedene Probleme und BUGS).







28.6 Reihen - Formate (row format)

MySQL kennt keine echten SQL VARCHAR Typen, dafür besitzt MySQL 3 verschiedene Möglichkeiten, Einträge zu speichern und VARCHAR zu emulieren.

Falls eine Tabelle keine VARCHAR, BLOB oder TEXT Spalten enthält, dann wird intern in der MySQL Datenbank eine feste Größe einer Spalte angenommen. Andernfalls, also wenn nur eine einzige Spalte von einem variablen Typ ist, dann wird eine dynamische Spaltengröße angenommen. CHAR und VARCHAR Spalten werden dann aus der Sicht der Anwendung identisch behandelt.

Das Format der Tabellen kann man mit dem SHELL-Befehl **isamchk -d ....** (das d steht für beschreibe die Tabelle) ausgeben lassen.

MySQL besitzt 3 verschiedene Tabellenformate, fest, dynamisch und compressed. Diese werden nun im einzelnen vorgestellt:

- **Fest** ist das Standardformat. Es wird verwendet, wenn die Tabelle keine VARCHAR, BLOB oder TEXT Spalten enthält. Die Vorteile sind, daß die Datenbank sehr schnell ist, leicht zu cachen, leicht zu reparieren (Im Falle eines Crash), benötigt keine ständige Reorganisation (isamchk), solange nicht Einträge in größerer Anzahl gelöscht werden, verbarucht aber auch gewöhnlich mehr Speicherplatz, als dynamische Tabellen (überflüssige Platzhalter).
- **Dynamisch** wird automatisch eingesetzt, sobald VARCHAR, BLOB oder TEXT Felder definiert werden, oder die Länge größer als 4 Bytes ist. Dynamische Felder besitzen einige Eigenheiten:
  - Alle Spaltenwerte werden auf dynamisch gesetzt.
  - Jeder Eintrag besitzt eine Bitmap, die anzeigt, welche Spalte leer ist. Hierzu werden doppelte Anführungszeichen oder 0 verwendet. Nicht leere Strings enthalten zuerst die Länge, danach folgt der Inhalt.
  - Dynamische Felder verbrauchen viel weniger Platz
  - Wenn Einträge größer werden, dann findet eine automatische Fragmentierung statt, die von Zeit zu Zeit mit **isamchk -r** bereinigt werden muß. Der Befehl **isamchk -ei tbl_name** zeigt Statistiken an.
  - Dynamische Tabellen sind schwieriger zu rekonstruieren, weil eventuell ein Teil der Fragmentierungsinformationen verlorengegengen ist.
- **Komprimierte Tabellen** sind read-only Tabellen, die zuvor mit dem **pack_isam** Utility komprimiert wurden. Alle registrierten User mit erweitertem E-Mail Support erhalten eine Kopie dieses Programmes. Diese Tabellen verbrauchen sehr wenig Speicherplatz. Jeder Eintrag wird separat komprimiert (wichtig zu wissen, da die Kompression insgesamt dann nicht so gut ist, als wenn man alle Einträge komprimieren würde, wie bei ZIP, also weg damit !). Blob und TEXT Spalten werden nicht komprimiert !!!







28.7 DELETE

```
DELETE [LOW_PRIORITY] FROM tbl_name
    [WHERE where_definition] [LIMIT rows]
```

DELETE löscht von tbl_name diejenigen Reihen, auf welche die Bedingung where_definition zutrifft, und gibt die Zahl der gelöschten Einträge zurück. Um alle Reihen zu löschen gibt man keine Bedingung mit an. Der Löschvorgang läuft dann viel schneller ab.

DROP INDEX löscht INDIZES

DROP Primary Key löscht Schlüssel

Diese Statements werden in Zusammenhang mit ALTER verwendet.







28.8 SELECT

```
SELECT [STRAIGHT_JOIN] [SQL_SMALL_RESULT] [DISTINCT | DISTINCTROW | ALL]
    select_expression,...
    [INTO OUTFILE 'file_name' export_options]
    [FROM table_references
        [WHERE where_definition]
        [GROUP BY col_name,...]
        [HAVING where_definition]
        [ORDER BY {unsigned_integer | col_name} [ASC | DESC] ,...]
        [LIMIT [offset,] rows]
        [PROCEDURE procedure_name] ]
```

SELECT wird eingesetzt, um Reihen aus einer oder mehreren Tabellen abzufragen. select_expression zeigt an, welche Spalten ausgewählt werden sollen. Das simpelste Bespiel:

```
mysql> SELECT 1 + 1;
        -> 2
```

Alle Schlüsslworte müssen in exakt dieser Reihenfolge angegeben werden. Der Ausdruck **HAVING** muß hierbei nach allen GROUP BY Ausdrücken, jedoch vor allen ORDER BY Ausdrücken stehen. Der SELECT Befehl kann zusammen mit dem Alias AS verwendet werden. Er kann mit ORDER BY und HAVING Ausdrücken verwendet werden:

```
mysql> select concat(last_name,', ',first_name) AS full_name
    from mytable ORDER BY full_name;
```

Die FROM table-references Ausdrücke zeigen an, daß Zeilen aus den Tabellen selektiert werden. Falls mehr als eine Tabelle angegeben wird, dann wird ein JOIN ausgeführt, siehe Kapitel JOIN. Der Befehl kann sich auf eine Spalte als col_name, tbl_name.col_name oder db_name.tbl_name.col_name beziehen. Eine Tabellenreferenz kann mit einem Alias versehen werden, wenn man tbl_name [AS] alias_name verwendet:

```
mysql> select t1.name, t2.salary from employee AS t1, info AS t2
           where t1.name = t2.name;
mysql> select t1.name, t2.salary from employee t1, info t2
           where t1.name = t2.name;
```

Ausgewählte Spalten dürfen für die Ausgabe sortiert werden. Hierfür benutzt man ORDER BY und GROUP BY Ausdrücke:

```
mysql> select college, region, seed from tournament
           ORDER BY region, seed;
mysql> select college, region AS r, seed AS s from tournament
           ORDER BY r, s;
```

```
mysql> select college, region, seed from tournament
           ORDER BY 2, 3;
```

Um in umgekehrter Reihenfolge zu sortieren, muß der Ausdruck DESC (descending) an den Namen der Spalte angefügt werden, zusammen mit dem Ausdruck ORDER BY. In Aufsteigender Reihenfolge wird mit dem Schlüsselwort ASC sortiert. Der Ausdruck HAVING kan sich auf irgendeine Spalte oder einen Alias beziehen, der in select_expression angegeben wurde. Er wird zuletzt angegeben. Benutzen Sie niemalös HAVING für Angaben, die in dem Ausdruck WHERE stehen sollten. Beispielsweise ist dieser Ausdruck verboten:

```
mysql> select col_name from tbl_name HAVING col_name > 0;
```

Stattdessen muß es so heißen:

```
mysql> select col_name from tbl_name WHERE col_name > 0;
```

In MySQL 3.22.5 oder höher ist auch folgende Schreibweise erlaubt:

```
mysql> select user,max(einkommen) from angestellte
           group by angestellte HAVING max(einkommen)>10000;
```

HAVING ist eine Art Nachselektierung des vorangegangenen SQL Statements. In unsererm Beispiel werden alle maximalen Löhne der Angestellten herausgesucht werden, allerdings werden alle unter 10.000 DM Einkommen nicht ausgegeben. In älteren Versionen muß dieses so formuliert werden:

```
mysql> select user,max(einkommen) AS sum from angestellte
           group by angestellte HAVING sum>10;
```

STRAIGHT_JOIN zwingt den Optimizer, Tabellen in der Reihenfolge zu verknüpfen, in der sie in dem Ausdruck FROM stehen. Falls der Optimizer Tabellen nicht optimal verknüpft, könnte dies die Abfrage beschleunigen.

SQL_SMALL_RESULT kann mit GROUP BY oder DISTINCT eingesetzt werden, um dem Optimizer mitzuteilen, daß das Ausgabeergebnis recht klein ist. In diesem Falle wird **MySQL** schnelle temporäre Tabellen anlegen, um das Ergebnis zu speichern, anstelle des Sortierens. Dieser Befehl ist eine **MySQL** spezifische Erweiterung und nicht in ANSI SQL92 enthalten.

Der Ausdruck LIMIT kann benutzt werden, um die Zahl der Reihen bei der Ausgabe zu begrenzen. LIMIT braucht zwei Zahlen als Argumente. Die erste Zahl bestimmt den Offset der ersten Reihe, die zweite Zahl die Zahl der Ausgaben. Der Offset beginnt mit 0, nicht wie sonst alles mit 1:

```
mysql> select * from table LIMIT 5,10;  # Zeilen 6-15 ausgeben
```

Ist nur ein Argument angegeben, wird angenommen, daß die Zahl der Reihen gemeint ist:

```
mysql> select * from table LIMIT 5;     # Gebe die ersten 5 Zeilen aus
```

Das bedeutet, daß LIMIT n identisch zu LIMIT 0,n ist. Insbesondere für Ausgaben im Browser über das Internet ist es wichtig, daß diese begrenzt werden. Ansonsten muß mit DoS Angriffen (Denial of Service = Außer Gefecht setzen) oder Überlastungen des Server gerechnet werden.

Der Befehl SELECT ... INTO OUTFILE 'file_name' bedeutet, daß SELECT die Ausgabe in eine Datei schreibt. Diese wird auf dem Server Host angelegt. Es können nur Dateien angelegt werden, die noch nicht existieren, damit nicht versehentlich die Datei **/etc/passwd** überschrieben wird.....

Es müssen auf dem Host zum Schreiben einer Datei privilegien zuvor vergeben worden sein (umask 022 in der Datei /etc/profiles).

INTO OUTFILE ist das Gegenstück zu LOAD DATA INFILE. Die Syntax kann aus einigen FIELDS und LINES Ausdrücken bestehen. Die Textdatei, in die ausgegeben wird, sind nur die folgenden Buchstaben mit einem Fluchtsymbol versehen (EXCAPED BY character). Es gibt hier die Ausdrücke FIELDS TERMINATED BY und LINES TERMINATED BY. ASCII 0 wird zu einem ESCAPED 0 und einer zusätzlichen 0 konvertiert, ein Tribut an die C-Syntax von Arrays.

Die FIELDS TERMINATED BY, EXCAPED BY oder LINES TERMINATED BY Buchstaben müssen so gespeichert werden, damit die Datei sauber wieder eingelesen werden kann. ASCII 0 wird von einigen Editoren besser verstanden.





28.9 JOIN

MySQL unterstützt die folgenden JOIN Ausdrücke in SELECT Statements:

```
table_reference, table_reference
table_reference [CROSS] JOIN table_reference
table_reference STRAIGHT_JOIN table_reference
table_reference LEFT [OUTER] JOIN table_reference ON conditional_expr
table_reference LEFT [OUTER] JOIN table_reference USING (column_list)
table_reference NATURAL LEFT [OUTER] JOIN table_reference
{ oj table_reference LEFT OUTER JOIN table_reference ON conditional_expr }
```

Die zuletzt angegebene LEFT OUTER JOIN Syntax existiert nur aus Kompatibilitätgründen zu ODBC. Eine Tabellen-Referenz kann mit einem Alias versehen werden: tbl_name AS alias_name oder tbl_name alias_name :

```
mysql> select t1.name, t2.salary from employee AS t1, info AS t2
           where t1.name = t2.name;
```

JOIN und , (Komma) sind von der Syntax her völlig äquivalent. Beide führen einen vollen JOIN Befehl zwischen zwei Tabellen aus. Normalerweise sollten Tabellen mit Hilfe der WHERE Bedingung verbunden werden. Der Ausdruck ON beschreibt ein kondizionales "Wenn es übereinstimmt", und kann mit dem Ausdruck WHERE verbunden werden. Wenn es keinen passenden Eintrag für die richtige Tabelle in einem LEFT JOIN gibt, wird eine Reihe, wo alle Spalten auf NULL gesetzt sind, für die rechte Tabelle verwendet. Es können somit Einträge gefunden werden, die kein entsprechendes Gegenstück in einer anderen Tabelle besitzten:

```
mysql> select table1.* from table1
           LEFT JOIN table2 ON table1.id=table2.id
           where table2.id is NULL;
```

Der einfachste JOIN ist der sogenannte "EQUI-JOIN". Ein Beispiel :

```
SELECT A.EineSpalte, B.EineAndereSpalte
FROM Tabelle1 AS A, Tabelle2 AS B WHERE A.EinWert = B.EinAndererWert;
```

Man kann ihn aber auch ganz anders schreiben, und die Ergebnismenge wird die gleiche sein, nämlich so :

```
SELECT A.EineSpalte, B.EineAndereSpalte
FROM Tabelle1 AS A JOIN Tabelle2 AS B
ON A.EinWert = B.EinAndererWert;
```

Wenn die Ergebnismenge die gleiche ist, wo liegt dann der Unterschied zwischen diesen beiden Formen ? Gibt es überhaupt einen Unterschied ?

Der Unterschied liegt in der Laufzeit. Im ersten Beispiel wird zuerst das kartesische Produkt aus beiden Tabellen gebildet (jede Zeile aus Tabelle1 wird mit jeder Zeile aus Tabelle2 verknüpft), und wenn beide Tabellen nur jeweils 100 Zeilen enthalten, sind das schon 10.000 Zeilen in der temporären Zwischentabelle. Erst dann werden die Zeilen gelöscht, die nicht die WHERE-Bedingung erfüllen.

Im zweiten Fall wird zuerst die Bedingung im ON-Teil geprüft und nur solche Zeilen in die Zwischentabelle übernommen, bei denen die Bedingung erfüllt ist. In dem Beispiel mit den je 100 Zeilen pro Tabelle sind das wahrscheinlich nicht mehr als 100 Zeilen. Das ist ein Faktor von 10 !

Dieses Beispiel findet alle Reihen in Tabelle 1 mit einem ID Wert, der nicht in Tabelle 2 enthalten ist. Das setzt voraus, daß table2.id ist als NOT NULL definiert worden.

Die der Ausdruck USING column_list benennt eine Liste von Spalten, die in beiden Tabellen existieren müssen. Ein Beispiel:

LEFT JOIN B USING (C1,C2,C3,...) ist identisch mit dem Ausdruck:

A.C1=B.C1 AND A.C2=B.C2 AND A.C3=B.C3,...

Das NATURAL LEFT JOIN zweier Tabellen ist identisch zu dem Ausdruck LEFT JOIN mit USING. Es werden dann alle Spalten genommen, die in beiden Tabellen exstieren. STRAIGHT JOIN ist identich mit JOIN, abgesehen davon, daß die linke Tabelle vor der rechten Tabelle ausgelesen wird. Dies kann dazu benutzt werden, wenn der JOIN Optimizer die Tabellen in der falschen Reihenfolge ausgibt (in seltenen Fällen). Ein paar Beispiele:

```
mysql> select * from table1,table2 where table1.id=table2.id;
mysql> select * from table1 LEFT JOIN table2 ON table1.id=table2.id;
mysql> select * from table1 LEFT JOIN table2 USING (id);
mysql> select * from table1 LEFT JOIN table2 ON table1.id=table2.id
           LEFT JOIN table3 ON table2.id=table3.id;
```

Hier nun ein Beispiel für ein bestimmtes Problem, welches sich nur mit Hilfe eines **LEFT JOINS** lösen läßt. Es gibt hier zwei Tabellen, **tabelle1** und **tabelle2**. Es sollen alle Elemente ausgegeben werden, die in **tabelle1**, aber nicht in **tabelle2** enthalten sind. In den Tabellen sind jeweils mitdestens eine Spalte namens **id** enthalten. Dieses Konstrukt würde bei Datenbanken, die das Kommando **SELECT ... MINUS SELECT ..**, also Subselects zulassen, in MySQL dasselbe bewirken:

```
SELECT table1.* from table1
LEFT JOIN table2 ON table1.id=table2.id
where table2.id is NULL
```

OUTER JOINs

Folgende Aufgabe : Sie wollen eine Preisliste drucken, und überall dort, wo es einen Lieferanten aus England gibt, soll dessen Name angedruckt werden. Bei den anderen Artikeln soll diese Ergebnis Spalte leer bleiben.

Ein einfacher JOIN reicht hier nicht mehr. Der Versuch, das Ergebnis so zu erzielen, wird scheitern:

```
SELECT A.ArtikelNr, B.Lieferant
FROM Artikel AS A, Lieferanten AS B WHERE A.LieferantID = B.LieferantID
AND B.Land = 'GB';
```

Dieses Statement macht nicht das, was es soll. Die Ergebnismenge enthält nur die Artikel mit englischen Lieferanten. Der Grund ist auch klar : nur bei diesen Artikeln ist die zweite WHERE-Bedingung hinter dem "AND" erfüllt. Wir brauchen also etwas anderes : den OUTER JOIN. Der sieht so aus :

```
SELECT A.ArtikelNr, B.Lieferant
FROM Artikel AS A LEFT OUTER
JOIN Lieferanten AS B ON A.LieferantID = B.LieferantIDWHERE B.Land = 'GB';
```

Bei einem OUTER JOIN gibt es immer eine Tabelle, die als erhaltene Tabelle bezeichnet wird; alle ihre Zeilen bleiben erhalten. Wenn es sich um einen LEFT OUTER JOIN handelt, ist die linke Tabelle (in unserem Beispiel Tabelle1) die erhaltene Tabelle; bei einem RIGHT OUTER JOIN ist es die rechte Tabelle (Tabelle2).

Die ON-Bedingung muß keine Gleichheits-Bedingung sein. Nehmen wir an, Sie wollen für jeden Tag die Anzahl der Bestellungen zeigen, die vor diesem Tag fällig waren, und alle Tage sollen im Ergebnis enthalten sein. Das erfordert wieder einen LEFT OUTER JOIN, und könnte so aussehen :

```
SELECT A.Tag, COUNT(B.BestellNr)
FROM Tage AS A LEFT OUTER JOIN
Bestellungen AS B ON A.Tag > B.FaelligkeitsDatum;
```

Firma Microsoft z.B., hält sich mal wieder nicht an Standards, und implementiert den OUTER JOIN so:

```
....  FROM Tabelle1 AS A, Tabelle2 AS B WHERE A.Code *= B.Code....
```

Tabelle1 ist diejenige Tabelle, die erhalten bleibt. Bei den EQUI-JOINS, bei denen die ON Bedingung die Gleichheit ausdrückt, funktioniert obiges Beispiel. Mit der Microsoft Syntax bei SQL Server 6.5 und 7.0 ist es leider nicht möglich.

Ein ganz konkretes Beispiel für die Problematik mit JOINS:

```
mysql> SELECT * FROM a;
+----+----------+
| id | ort      |
+----+----------+
|  1 | passau   |
|  2 | münchen  |
|  3 | salzburg |
|  4 | passau   |
+----+----------+
4 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM b;
+----+
| id |
+----+
|  3 |
|  4 |
+----+
2 rows in set (0.00 sec)
```

In Tabelle a steht also, welche Personen in welchem Ort sind.

In Tabelle b steht, welche Personen einer Niederlassung angehören.

Nun sehen wir mal Schritt für Schritt, was bei "a.id != b.id" genau passiert. (Sortierungen mit ORDER BY mache ich jeweils nur der Übersichtlichkeit halber; an der Ergenismenge selbst ändert sich dadurch ja nichts.)

Erstmal das komplette Kreuzprodukt, das man bekommt, wenn man einen JOIN ohne Bedingung durchführt ("p_in_o" steht für "Person in Ort", "p_in_n" für "Person in Niederlassung"):

```
mysql> SELECT a.ort, a.id AS p_in_o, b.id AS p_in_n
   --> FROM a, b
   --> ORDER BY p_in_o, p_in_n;
+----------+--------+--------+
| ort      | p_in_o | p_in_n |
+----------+--------+--------+
| passau   |      1 |      3 |
| passau   |      1 |      4 |
| münchen  |      2 |      3 |
| münchen  |      2 |      4 |
| salzburg |      3 |      3 |    <-- Gleichheit
| salzburg |      3 |      4 |
| passau   |      4 |      3 |
| passau   |      4 |      4 |    <-- Gleichheit
+----------+--------+--------+
8 rows in set (0.00 sec)
```

Das kombiniert also alle Tupel aus Tabelle a mit allen Tupeln aus Tabelle b.

Die markierten Zeilen haben die Bedeutung: diese Person wohnt in diesem Ort und ist in einer Niederlassung.

Die anderen Zeilen haben die Bedeutung: die Person p_in_o wohnt in diesem Ort und die andere Person p_in_n ist in einer Niederlassung. Das ergibt also keine Aussage über eine Person!

Mit dem Zusatz "WHERE a.id != b.id" fallen jetzt die Zeilen weg, wo z.B. "a.id = b.id" wäre:

```
mysql> SELECT a.ort, a.id AS p_in_o, b.id AS p_in_n
   --> FROM a, b
   --> WHERE a.id != b.id
   --> ORDER BY a.id, b.id;
+----------+--------+--------+
| ort      | p_in_o | p_in_n |
+----------+--------+--------+
| passau   |      1 |      3 |
| passau   |      1 |      4 |
| münchen  |      2 |      3 |
| münchen  |      2 |      4 |
| salzburg |      3 |      4 |
| passau   |      4 |      3 |
+----------+--------+--------+
6 rows in set (0.00 sec)
```

Diese Tabelle enthält in Spalte p_in_o NICHT nur die Leute, die in keiner Niederlassung sind .....

Und selbst bei den Personen in Spalte p_in_o, die tatsächlich in keiner Niederlassung sind, stimmen die Anzahlen nicht, die man per COUNT(*) ... GROUP BY ... ermitteln kann, da z.B. Person 1 zweimal erscheint. Gruppiert man nun nach dem Ort, würden sich z.B. für "passau" hier zusammen dann drei Zeilen ergeben. Korrekt wäre aber nur die Ausgabe einer Zeile !!!!!

Wie macht man es korrekt ? In Wirklichkeit ist die Denkweise falsch. Es müssen alle Personen gesucht werden, die nicht in der Tabelle b enthalten sind, also keiner Niederlassung angehören:

Erster Schritt: Personen, die einer Niederlassung angehören:

```
mysql> SELECT a.ort, a.id AS p_in_o, b.id AS p_in_n
   --> FROM a, b
   --> WHERE a.id = b.id
   --> ORDER BY p_in_o, p_in_n;
+----------+--------+--------+
| ort      | p_in_o | p_in_n |
+----------+--------+--------+
| salzburg |      3 |      3 |
| passau   |      4 |      4 |
+----------+--------+--------+
2 rows in set (0.00 sec)
```

Es sind also genau zwei Zeilen, die gleiche Einträge haben. Dummerweise fallen aber so die anderen Personen aus Tabelle a heraus. Genau diese hätten wir aber gerne ebenfalls noch mit ausgegeben. Hier hilft der **LEFT JOIN** !

Er bewirkt, daß auch die Werten aus der linken Tabelle ausgegeben werden, zu denen es keinen Join-Partner aus der anderen Tabelle gibt:

```
mysql> SELECT a.ort, a.id AS p_in_o, b.id AS p_in_n
   --> FROM a LEFT JOIN b
   -->      ON a.id = b.id
   --> ORDER BY p_in_o, p_in_n;
+----------+--------+--------+
| ort      | p_in_o | p_in_n |
+----------+--------+--------+
| passau   |      1 |   NULL |
| münchen  |      2 |   NULL |
| salzburg |      3 |      3 |
```

```
| passau   |      4 |      4 |
+----------+--------+--------+
4 rows in set (0.00 sec)
```

Nun dürfte klar sein, daß die Personen, die in keiner Niederlassung sind, man mit Hilfe des Zusatzes: **WHERE b.id IS NULL** ermitteln kann.

Im WHERE-Teil kann man leider nicht mit Alias-Namen arbeiten.....

```
mysql> SELECT a.ort, a.id AS p_in_o, b.id AS p_in_n
    --> FROM a LEFT JOIN b
    -->      ON a.id = b.id
    --> WHERE b.id IS NULL
    --> ORDER BY p_in_o, p_in_n;
+---------+--------+--------+
| ort     | p_in_o | p_in_n |
+---------+--------+--------+
| passau  |      1 |   NULL |
| münchen |      2 |   NULL |
+---------+--------+--------+
2 rows in set (0.00 sec)
```

Hier nun das fertige Statement:

```
SELECT a.ort, COUNT(*) AS anzahl
FROM person AS a LEFT JOIN rel_niederlassung_person AS b
ON a.id = b.id
WHERE b.id IS NULL
GROUP BY a.ort;
```

Somit dürfte auch klar sein, wie man einen "RIGHT JOIN" mit MySQL realisieren kann .... nämlich einfach durch Vertauschung der linken und rechten Tabelle .....






28.10 INSERT

```
    INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
        [INTO] tbl_name [(col_name,...)]
        VALUES (expression,...),(...),...
oder  INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
        [INTO] tbl_name [(col_name,...)]
        SELECT ...
oder  INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
        [INTO] tbl_name
        SET col_name=expression, col_name=expression, ...
```

INSERT fürgt neue Reihen in eine Tabelle ein. Die INSERT ... VALUES Ausdruck fügt Reihen ein, unter Verwendung genau angegeber Zahlen. Das INSERT ... SELECT Ausdruck fügt eine Reihe ein, die aus einer anderen Tabelle ausgewählt wurden. Der Ausdruck INSERT ... VALUES mit der Angabe einer mehrfachen Werteliste wird ab **MySQL** 3.22.5 unterstützt. Der Ausdruck col_name=expression wird ab **MySQL** 3.22.10 unterstützt.

tbl_name ist die Tabelle, in welche die Reihen eingefügt werden sollen. Die Zusatzangabe der Liste col_name oder der Ausdruck SET zeigt an, auf welche Spalten der Ausdruck wirkt.

Wenn keine Spalten Liste bei INSERT ... VALUES oder INSERT ... SELECT, angegeben ist, dann müssen die Spalten in der VALUES() Liste oder im SELCT Statement angegeben werden. Allen Spalten, denen kein Wert zugewiesen wird, fallen auf den Defaultwert zurück. Wenn, zum Beispiel, eine Spaltenliste angegeben wird, die nicht alle Spalten in der Tabelle aufzählt, dann werden die ungenannten Spalten auf ihre Defaultwerte gesetzt. Der Default Wert ist bei CREATE angegeben. Ein Ausdruck kann sich aber auch auf alle Spalten beziehen, die etwas früher in einer Liste genannt wurden:

```
mysql> INSERT INTO tbl_name (col1,col2) VALUES(15,col1*2);
```

Dieser Ausdruck ist nicht erlaubt:

```
mysql> INSERT INTO tbl_name (col1,col2) VALUES(col2*2,15);
```

Wenn das Schlüsselwort **LOW_PRIORITY** angegeben wird, dann wird die Ausführung des **INSERT** Befehls solange verzögert, bis keine Abfrage mehr stattfindet. Wenn das Schlüsselwort **IGNORE** in einem **INSERT** Statement mit vielen Reihen angegeben wird, werden alle Reihen ignoriert, die einen existerenden **PRIMARY** oder **UNIQUE** Schlüsseleintrag enthalten. Wenn IGNORE nicht angegeben wird, bricht der INSERT genau dann ab, wenn dort eine Reihe ist, die

einen Schlüsselwert gerade dupliziert. Mit Hilfe des Befehls **mysql_info()** kann abgefragt werden, wieviele Reihen in die Tabelle ingefügt worden sind. Wenn **MySQL** so konfiguriert wurde, daß die Option **DONT_USE_DEFAULT_FIELDS** aktiviert wurde, erzeugen **INSERT** Statements eine Fehlermeldung, wenn speziell Werte für alle Spalten angegeben werden, die einen NON-NULL Wert erwarten.

Die folgenden Zusatzbedingungen für ein **INSERT INTO ... SELECT** Statement müssen erfüllt sein:

- Eine **Abfrage** darf **kein ORDER BY** enthalten
- Die Zieltabelle des INSERT Befehls darf nicht in FROM des SELECT Teils der Abfrage enthalten sein. Es ist unter ANSI SQL verboten, ein und dieselbe Tabelle für lesen und INSERT zu verwenden.

Wenn INSERT ... SELECT oder ein INSERT ... VALUES Statement mit vielfachen Wertelisten verwendet wird, dann kan die C API Funktion **mysql_info()** verwendet werden, um Informationen über die Abfrage zu erhalten. Das Format lautet wie folgt:

```
Records: 100 Duplicates: 0 Warnings: 0
```

Duplicates zeigt die Zahl der Reihen an, welche nicht eingefügt werden konnten, weil sie einen eindeutigen Index Wert kopieren würden. Warnings zeigt die Zahl der Versuche, Spaltenwerte einzufügen, an. Warnungen können aber auch unter folgenden Bedingungen auftreten:

- Einfügen einer NULL in eine Spalte, die als NOT NULL definiert wurde. Es wird der Defaultwert angenommen.
- Setzen einer numerischen Spalte auf einen Wert, der außerhalb des Spalten Limits liegt.
- Setzen einer Numerischen Spalte auf einen Wert, wie z.B. "10.34 a". Der Zahlenmüll wird entfernt und es wird ein "a", also ein Buchstabe als numerischer Wert für die Spalte genommen. Der Wert wird auf 0 gesetzt.
- Einfügen eines Strings in eine CHAR, VARCHAR, TEXT oder BLOB - Spalte, die die maximale Länge der Spalte überschreitet. Der String wird beim Maximum abgeschnitten.
- Einfügen eines Wertes in eine Datum oder Zeit-Spalte, die nicht als solche definiert wurde. Der Wert wird auf NULL gesetzt.

Die Option DELAYED für das INSERT Statement ist eine **MySQL** spezifische Option, die sehr sinvoll ist, wenn es Clients gibt, die nicht auf die Beendigung eines INSERT warten können. Das kann z.B. passieren, wenn **MySQL** für Logging verwendet wird, und gleichzeitig SELECT Statements laufen müssen, die sehr lange dauern. DELAYED wurde in **MySQL** 3.22.15 eingeführt und ist eine Erweiterung gegenüber dem ANSI SQL92 Standard.

Ein weiterer Vorteil von INSERT DELAYED ist, daß INSERTS von vielen Clients gebündelt werden, um in einem Vorgang geschrieben zu werden. Diese Vorgehensweise ist viel schneller, als einzelne INSERTS durchzuführen.

Die gequeuten Reihen werden immer nur im RAM gehalten, bis sie in die Tabelle eingefügt werden. Das bedeutet, wenn der **mysqld** Dämon gekillt wird, beispielsweise mit **kill -9**, dann sind die Daten verloren.






28.11 REPLACE

```
    REPLACE [LOW_PRIORITY | DELAYED]
        [INTO] tbl_name [(col_name,...)]
        VALUES (expression,...)
oder  REPLACE [LOW_PRIORITY | DELAYED]
        [INTO] tbl_name [(col_name,...)]
        SELECT ...
oder  REPLACE [LOW_PRIORITY | DELAYED]
        [INTO] tbl_name
        SET col_name=expression, col_name=expression,...
```

REPLACE arbeitet genauso wie INSERT, mit der Ausnahme, daß wenn ein alter Eintrag in der Tabelle denselben Wert hat, wie der neue Eintrag in einem eindeutigen INDEX, daß der alte Eintrag aus der Tabelle gelöscht wird, bevor der neue eingefügt wird.






28.12 LOAD DATA INFILE

```
LOAD DATA [LOCAL] INFILE 'file_name.txt' [REPLACE | IGNORE]
    INTO TABLE tbl_name
    [FIELDS
        [TERMINATED BY '\t']
        [OPTIONALLY] ENCLOSED BY "]
        [ESCAPED BY '\\' ]]
    [LINES TERMINATED BY '\n']
    [IGNORE number LINES]
    [(col_name,...)]
```

Der Befehl **LOAD DATA INFILE** liest Reihen aus einer Textdatei in eine Tabelle mit einer sehr hohen Geschwindigkeit. Wenn das Schlüsselwort LOCAL angegeben wurde, dann wird die Datei von dem Client Host gelesen. Falls nicht, dann muß sich die Datei auf dem Server befinden. (LOCAL ist ab Version 3.22.6 verfügbar)

Viel einfacher ist es jedoch, mit Hilfe von Administrationswerkzeugen Datenbanken einzulesen oder zu konvertieren. Diese finden Sie im Kapitel Einsteiger Tutorial LINUX MySQL Server, nebenher gesagt....

Aus Sicherheitsgründen muß, wenn eine Textdatei auf dem Server eingelesen wird, diese entweder im Datenbankverzeichnis liegen, oder allgemein lesbar sein. Um also den Befehl LOAD DATA INFILE ausführen zu können, muß man die Datei Privilegien (file privilege) besitzen. Meh hierzu im Kapitel Datei Privilegien.

Der Befehl **LOCAL** wird ein wenig langsamer sein, als der direkte Zugriff des Servers, da die Daten erst über das Netzwerk übertragen werden müssen. Andererseits sind in diesem Falle keinerlei Privilegien notwendig.

Datenfiles können auch mit Hilfe des mysqlimport Utility eingelesen werden. Es arbeitet identisch zu dem Befehl LOAD DATA INFILE auf dem Server. Die LOCAL Option veranlaßt den Server, die Daten vom Client über das Netzwerk zu laden. Die Option --compress beim Befehl mysqlimport veranlaßt Client und Server, die Daten vor der Übermittlung zu komprimieren.

networks if the client and server support the compressed protocol.

Für das Auffinden der Dateien auf dem Server Host gelten folgende Regeln:

- Absolute Pfadnamen können angegeben werden
- Bei relativen Pfadnamen gilt als Referenz das Datenverzeichnis des SQL Dämons
- Bei der Angabe von ./daten.txt wird vom Datenverzeichnis des Servers gelesen, bei der Angabe daten.txt allein wird aus dem Datenbankverzeichnis der aktuellen Datenbank gelesen

Folgendes Statement liest aus dem Datenbankverzeichnis für db1, nicht db2:

```
mysql> USE db1;
mysql> LOAD DATA INFILE "./data.txt" INTO TABLE db2.my_table;
```

Die Schlüsselworte REPLACE und IGNORE kontrollieren das Handling der eingehenden Einträge, die exisiterende Einträge mit eindeutigen KEY VALUES kopieren. Wenn REPLACE angegeben wird, ersetzen neue Einträge alte, die dieselben KEY VALUES besitzen. Wenn IGNORE angegeben wird, dann werden diese Einträge übergangen. Ohne Angabe wird ein Fehler ausgegeben, wenn ein doppelter KEY VALUE auftritt.

Wenn Daten von einer lokalen Datei (auf dem Client) mit Hilfe des LOCAL Schlüsselwortes eingeladen werden, kann der Server die Übertragung der Daten nicht abbrechen, also handelt dieser, als wenn der Befehl zum Laden mit IGNORE gesterete worden wäre.

LOAD DATA INFILE ist das Gegenstück zu **SELECT ... INTO OUTFILE**. Siehe hierzu auch das Kapitel SELECT. Um Daten von einer Datenbank in eine Datei zu schreiben und wieder einzulesen, sind die Befehle SELECT ... INTO OUTFILE und LOAD DATA INFILE zu verwenden. Die Syntax für die Optionen FIELDS und LINES sind für beide Statements identisch und optional. Der Befehl FIELDS muß vor dem Befehl LINES stehen.

Zusammen mit dem Ausdruck **FIELDS** kann optional der Unterausdruck (TERMINATED BY, ENCOLSED BY und ESCAPED BY verwendet werden. Zumindest einer dieser Ausdrücke muß angewendet werden.

Wenn der Ausdruck FIELDS nicht angegeben wird, wird der Befehl so interpretiert:

```
FIELDS TERMINATED BY '\t' ENCLOSED BY '' ESCAPED BY '\\'
```

Wenn kein LINES Ausdruck verwendet wird, wird folgender Befehl angenommen:

```
LINES TERMINATED BY '\n'
```

Anders gesagt, der Ausdruck **LOAD DATA INFILE** wird dann so ausgeführt:

- Schaue nach Zeilenenden bei \newline
- Breche Zeilen in Felder um bei \tab
- Erwarte nicht, daß Felder durch Anführungszeichen irgendeiner Art gekennzeichnet sind, z.B. mit \' .

Der Ausdruck SELECT ... INTO OUTFILE arbeitet folgendermaßen:

- Schreibe \tab zwischen Felder
- Keine Felder mit Anführungszeichen einrahmen
- Benutze \' um Instanzen von \tab, \newline oder \' zu unerdrücken
- Schreibe \newline am Ende einer Zeile
- Schreibe ein \ als \\

```
mysql> LOAD DATA INFILE "/tmp/file_name" into table test IGNORE 1 LINES;
```

Wenn SELECT ... INTO OUTFILE ausgeführt wird und direkt danach der Befehl LOAD DATA INFILE, müssen die Handling Optionen genau identisch sein, andern falls werden die Inhalte nicht korrekt interpretiert. Ein Beispiel:

```
mysql> SELECT * FROM table1 INTO OUTFILE 'data.txt'
           FIELDS TERMINATED BY ','
           FROM ...
```

Um eine durch Komma in den Feldern separierte Datei wieder einzuladen, muß also korrekterweise angegeben werden:

```
mysql> LOAD DATA INFILE 'data.txt' INTO TABLE table2
           FIELDS TERMINATED BY ',';
```

Folgender Befehl würde unweigerlich scheitern:

```
mysql> LOAD DATA INFILE 'data.txt' INTO TABLE table2
           FIELDS TERMINATED BY '\t';
```

Jede Zeile würde als einziges Feld interpretiert werden.

LOAD DATA INFILE kann benutzt werden, um Dateien von externen Quellen zu laden. Z.B. eine Datei im DBASE Format hat die Felder durch Kommas getrennt ,und diese in doppelte Anführungszeichen eingepackt. Wenn die Zeilen durch \newline getrennt sind, dann wird folgender Befehl die Datei einladen:

```
mysql> LOAD DATA INFILE 'data.txt' INTO TABLE tbl_name
           FIELDS TERMINATED BY ',' ENCLOSED BY '"'
           LINES TERMINATED BY '\n';
```

Jede der Felder oder Line Handling Optionen kann ein leeren String erkennen. Wenn dieser nicht leer ist, müssen die Werte für **FIELDS [OPTIONALLY] ELCLOSED BY** und **FIELDS ESCAPED BY** mehr als einen Buchstaben enthalten. Z.B. muß man, um eine Zeilen zu schreiben, die durch ein \cr terminiert sind, den Ausdruck TERMINATED BY '\r\n' verwenden. Beipiel:

```
"1","a string","100.20"
"2","a string containing a , comma","102.20"
"3","a string containing a \" quote","102.20"
"4","a string containing a \", quote and comma","102.20"
```

Wenn OPTIONALLY angegeben wird, wird der ENCLOSE BY Buchstabe nur benutzt, um CHAR und VARCHAR Felder zu trennen:

```
1,"a string",100.20
2,"a string containing a , comma",102.20
3,"a string containing a \" quote",102.20
4,"a string containing a \", quote and comma",102.20
```

Sie sollten berücksichtigen, daß wenn immer der Ausdruck ENCLOSED BY Buchstabe in einem Feld vorkommt, diesem der ESCAPED BY Buchstabe vorangestellt werden muß. Beachten Sie hierzu das folgende Beispiel. Beispiel 4 (letzte Zeile) besitzt die richtige Syntax, auch wenn es so aussieht, als wenn das Anführungszeichen vor dem Komma das Feld beenden würde. Der Parser reagiert auf einen String, der sich aus den beiden ENCOSED BY und TERMINITATED BY Buchstaben zusammensetzt, mit einer Sonderbehandlung. Er schließt dann das vorhergehende Feld dann noch nicht ab, sondern merkt sich das Komma als Buchstabe, der in des String gehört.

Beispiel Zwei würde einen Fehler verursachen, weil der Parser das Komma als Trennfeld betrachen würde. Danach würden ein Gemisch aus String und Zahl folgen, welches zu einem Fehler führt. Bei Beispiel drei führt das zu einem Fehler, da ein Leerzeigen auf den ENCLOSED BY Buchstaben folgt. Hier endet für den Parser das Feld. Danach folgen keine reinen Zahlen mehr.

```
1,"a string",100.20
```

```
2,"a string containing a , comma",102.20
3,"a string containing a " quote",102.20
4,"a string containing a ", quote and comma",102.20
```

Hier ein Beispiel mit ENCLOSED BY:

```
"The ""BIG"" boss"  -> The "BIG" boss
The "BIG" boss      -> The "BIG" boss
The ""BIG"" boss    -> The ""BIG"" boss
```

Das folgende Beipiel lädt alle Spalten der Tabelle persondata:

```
mysql> LOAD DATA INFILE 'persondata.txt' INTO TABLE persondata;
```

Es könne aber auch nur einige Spalten der Tabelle geladen werden:

```
mysql> LOAD DATA INFILE 'persondata.txt'
           INTO TABLE persondata (col1,col2,...);
```

Wenn eine Reihe zu wenig Felder hat, werden die Spalten, die leerbleiben, alle auf ihre Defaultwerte gesetzt, meist 0.

Alle Daten beim Einlesen werden als Strings behandelt. Es können also keine numerischen Werte für ENUM oder SET Spalten, so wie es mit dem INSERT Statment funktioniert. Alle ENUM und SET Werte müssen als Strings angegeben werden.






28.13 UPDATE

```
UPDATE [LOW_PRIORITY] tbl_name SET col_name1=expr1,col_name2=expr2,...
    [WHERE where_definition]
```

UPDATE setzt neue Werte in Spalten von existierenden Tabellen ein. Der Ausdruck SET zeigt an, welche Spalten modifiziert werden sollen. Die Option WHERE, gibt die Reihe an, andernfalls werden alle Reihen modifiziert.

Bei der Angabe des Schlüsselwortes LOW_PRIORITY wird die Ausführung von UPDATE solange verzögert, bis kein Client mehr lesend zugreift.

Das folgende Beispiel erhöht den Wert einer ganzen Spalte um eins:

```
mysql> UPDATE persondata SET age=age+1;
```

UPDATE Zuweisungen werden von links nach rechts ausgeführt. Folgendes Beispiel verdoppelt zuerst die Werte, danach erste werde diese um eins erhöht:

```
mysql> UPDATE persondata SET age=age*2, age=age+1;
```

UPDATE gibt die Zahl der Reihen zurück, die aktuell verändert wurden. Mit mysqlinfo() können diese Rückgabe-Werte angezeigt werden.







28.14 USE

```
USE db_name
```

Das Statement USE db_name teilt **MySQL** mit, daß die Datenbank db_name die Standarddatenbank nun ist. Alle Befehle wirken bis auf Wiederruf auf diese Datenbank:

```
mysql> USE db1;
mysql> SELECT count(*) FROM mytable;      # select von db1.mytable
mysql> USE db2;
mysql> SELECT count(*) FROM mytable;      # select von db2.mytable
```

Wenn eine bestimmte Datenbank angesprochen wird, wird der Befehl USE übergangen:

```
mysql> USE db1;
mysql> SELECT author_name,editor_name FROM author,db2.editor
           WHERE author.editor_id = db2.editor.editor_id;
```

Der Ausdruck USE wurde aus Gründen der Kompatibilität zur SYBASE (und auch Microsoft) Datenbanke eingeführt.







28.15 FLUSH

```
FLUSH flush_option [,flush_option]
```

FLUSH löscht die internen Cache von **MySQL**. Um FLUSH ausführen zu dürfen, muß das RELOAD - Privileg vergeben sein.

Die Optionen bei FLUSH können folgende sein:

- HOSTS löscht die Host Cache Tables. Falls der Server irgendwie wegen TCP/IP Störungen hängen sollte, reicht FLUSH HOSTS aus.
- LOGS Schließt und Öffnet die Log-Dateien.
- PRIVILEDGES Lädt die Privilegien aus den GRANT Tabellen neu ein.
- TABLES schließt alle Tabellen.
- STATUS setzt alle Statusvariablen auf Null

Alle Befehle können auch mit Hilfe des mysqladmin Utility abgesetzt werden.







28.16 KILL

```
KILL thread_id
```

Wenn das PROCESS PRIVILEDGE vergeben ist, können so Threads allgemein beendet werden. , andernfalls können nur die eigenen Prozesse beendet werden.

Der Befehl SHOW PROCESSLIST zeigt alle Threads an.






28.17 SHOW

```
     SHOW DATABASES [LIKE wild]
oder SHOW TABLES [FROM db_name] [LIKE wild]
oder SHOW COLUMNS FROM tbl_name [FROM db_name] [LIKE wild]
oder SHOW INDEX FROM tbl_name [FROM db_name]
oder SHOW STATUS
oder SHOW VARIABLES [LIKE wild]
oder SHOW PROCESSLIST
oder SHOW TABLE STATUS [FROM db_name] [LIKE wild]
```

SHOW zeigt Informationen über Datenbanken, Tabellen, Spalten oder den Server an. Wenn die Option LIKE benutzt wird, können Wildcards angegeben werden. Folgende Befehle sind identisch:

```
mysql> SHOW INDEX FROM mytable FROM mydb;
mysql> SHOW INDEX FROM mydb.mytable;
```

SHOW DATABASES listet alle Datenbanken auf dem **MySQL**. Alternativ kann auch das Commandline Utility mysqlshow verwendet werden. Wer unzureichende Privilegien hat, der sieht keine fremden Tabellen.

Der Befehl **SHOW TABLE STATUS** (neu in 3.23) arbeitet wie SHOW STATUS, liefert nur wesentlich mehr Informationen über jede Tabelle, alternativ kann auch der Befehl mysqlshow --status db_name in der Commandline eingegeben werden:

- Name Name der Tabelle
- Type Type der Tabelle (NISAM, MyISAM or HEAP)
- Rows Zahl der Reihen
- Avg_row_length Durchschnittliche Länge der Reihen
- Data_length Länge des Datenfile
- Max_data_length Maximale Länge des Datenfile
- Index_length Länge der INDEX Datei
- Data_free Zahl der reservierten Bytes
- Auto_increment Nächster Autoincrement Wert
- Create_time Wann die Tabelle angelegt wurde
- Update_time Datum und Zeit des letzten Updates
- Check_time Uhrzeit des letzten Checks einer Tabelle
- Create_min_rows Die "min_rows" Option beim Anlegen einer Tabelle
- Create_max_rows Die "max_rows" Option beim Anlegen einer Tabelle
- Create_avg_row_length Die durchschnittliche Länge einer Reihe Option beim Anlegen einer Tabelle

- Comment Der Kommentar bei anlegen einer Tabelle

Die Befehle SHOW FIELDS sind identisch zu SHOW COLUMNS und SHOW KEYS ist identisch zu SHOW INDEX. Alternativ kann auch mysqlshow -k db_name tbl_name eingegeben werden.

SHOW INDEX gibt eine Index Information in einem Format zurück, welches an den Aufruf von SQLStatistics in ODBC erinnert. Folgende Spalten werden zurückgegeben:

- Table Name der Tabelle
- Non_unique 0 wenn der Index keine Duplikate enthalten darf.
- Key_name Name des Index
- Seq_in_index Spalten Sequenznummer, start bei 1.
- Column_name Spalten Name.
- Collation Wie wird die Spalte sortiert. In MySQL ist A (Ascending) oder NULL (unsortiert) möglich.
- Cardinality Zahl der eindeutigen Werte in einem Index. Update durch isamchk -a.
- Sub_part Zahl der indizierten Buchstaben, wenn die Spalte nur teilweise indiziert wird. NULL, wenn der ganze Schlüssel indiziert ist.

SHOW STATUS zeigt alle Status Informationen an:

```
+---------------------------+--------+
| Variable_name             | Value  |
+---------------------------+--------+
| Aborted_clients           | 0      |
| Aborted_connects          | 0      |
| Created_tmp_tables        | 0      |
| Delayed_insert_threads    | 0      |
| Delayed_writes            | 0      |
| Delayed_errors            | 0      |
| Flush_commands            | 2      |
| Handler_delete            | 2      |
| Handler_read_first        | 0      |
| Handler_read_key          | 1      |
| Handler_read_next         | 0      |
| Handler_read_rnd          | 35     |
| Handler_update            | 0      |
| Handler_write             | 2      |
| Key_blocks_used           | 0      |
| Key_read_requests         | 0      |
| Key_reads                 | 0      |
| Key_write_requests        | 0      |
| Key_writes                | 0      |
| Max_used_connections      | 1      |
| Not_flushed_key_blocks    | 0      |
| Not_flushed_delayed_rows  | 0      |
| Open_tables               | 1      |
| Open_files                | 2      |
```

```
| Open_streams              | 0      |
| Opened_tables             | 11     |
| Questions                 | 14     |
| Running_threads           | 1      |
| Slow_queries              | 0      |
| Uptime                    | 149111 |
+---------------------------+--------+
```

Ein paar Probleme, die ententuell auftauchen könnten:

Wenn die Zahl bei Opened_tables zu groß ist, dann ist die table_cache Variable zu klein.

Wenn key_reads zu groß ist, dann ist der key_cache zu klein.

Wenn Handler_read_rnd zu groß ist, dann muß **MySQL** zuviele Abfragen über ganze Tabellen durchführen. Dies ist ein Zeichen dafür, daß JOINS die Schlüssel nicht korrekt benutzen.

Der Befehl SHOW VARIABLES zeigt einige MySQL Systemvariablen an:

```
+-------------------------+----------------------------+
| Variable_name           | Value                      |
+-------------------------+----------------------------+
| back_log                | 5                          |
| connect_timeout         | 5                          |
| basedir                 | /my/monty/                 |
| datadir                 | /my/monty/data/            |
| delayed_insert_limit    | 100                        |
| delayed_insert_timeout  | 300                        |
| delayed_queue_size      | 1000                       |
| join_buffer_size        | 131072                     |
| flush_time              | 0                          |
| key_buffer_size         | 1048540                    |
| language                | /my/monty/share/english/   |
| log                     | OFF                        |
| log_update              | OFF                        |
| long_query_time         | 10                         |
| low_priority_updates    | OFF                        |
| max_allowed_packet      | 1048576                    |
| max_connections         | 100                        |
| max_connect_errors      | 10                         |
| max_delayed_threads     | 20                         |
| max_heap_table_size     | 16777216                   |
| max_join_size           | 4294967295                 |
| max_sort_length         | 1024                       |
| max_tmp_tables          | 32                         |
| net_buffer_length       | 16384                      |
| port                    | 3306                       |
| protocol-version        | 10                         |
| record_buffer           | 131072                     |
| skip_locking            | ON                         |
```

```
| socket                  | /tmp/mysql.sock          |
| sort_buffer             | 2097116                  |
| table_cache             | 64                       |
| thread_stack            | 131072                   |
| tmp_table_size          | 1048576                  |
| tmpdir                  | /machine/tmp/            |
| version                 | 3.23.0-alpha-debug       |
| wait_timeout            | 28800                    |
+-------------------------+--------------------------+
```

Siehe auch Kapitel EXPLAIN (Tuning der Geschwindigkeit)






28.18 EXPLAIN (Tuning der Geschwindigkeit)

Vorab ein wirklich einfaches Beispiel der Komplexität falscher Abfragen:

```
Tabelle 1

Zeile   Bemerkungen
==================
Zeile 1 Tabelle 1
Zeile 2 Tabelle 1
Zeile 3 Tabelle 1
Zeile 4 Tabelle 1
Zeile 5 Tabelle 1
Zeile 6 Tabelle 1


Tabelle 2

Zeile   Bemerkungen
==================
Zeile 1 Tabelle 2
Zeile 2 Tabelle 2
Zeile 3 Tabelle 2
Zeile 4 Tabelle 2
Zeile 5 Tabelle 2
Zeile 6 Tabelle 2
```

Wir verknüpfen nun Tabelle 1 mit Tabelle 2:

```
SELECT * FROM TABELLE1, TABELLE2

ZEILE BEMERKUNGEN ZEILE BEMERKUNGEN
==================================
Zeile 1 Tabelle 1 Zeile 1 Tabelle 2
Zeile 1 Tabelle 1 Zeile 2 Tabelle 2
Zeile 1 Tabelle 1 Zeile 3 Tabelle 2
Zeile 1 Tabelle 1 Zeile 4 Tabelle 2
Zeile 1 Tabelle 1 Zeile 5 Tabelle 2
Zeile 1 Tabelle 1 Zeile 6 Tabelle 2
Zeile 2 Tabelle 1 Zeile 1 Tabelle 2
Zeile 2 Tabelle 1 Zeile 2 Tabelle 2
Zeile 2 Tabelle 1 Zeile 3 Tabelle 2
Zeile 2 Tabelle 1 Zeile 4 Tabelle 2
Zeile 2 Tabelle 1 Zeile 5 Tabelle 2
Zeile 2 Tabelle 1 Zeile 6 Tabelle 2
Zeile 3 Tabelle 1 Zeile 1 Tabelle 2
Zeile 3 Tabelle 1 Zeile 2 Tabelle 2
Zeile 3 Tabelle 1 Zeile 3 Tabelle 2
Zeile 3 Tabelle 1 Zeile 4 Tabelle 2
Zeile 3 Tabelle 1 Zeile 5 Tabelle 2
Zeile 3 Tabelle 1 Zeile 6 Tabelle 2
Zeile 4 Tabelle 1 Zeile 1 Tabelle 2
```

```
Zeile 4 Tabelle 1 Zeile 2 Tabelle 2
Zeile 4 Tabelle 1 Zeile 3 Tabelle 2
Zeile 4 Tabelle 1 Zeile 4 Tabelle 2
Zeile 4 Tabelle 1 Zeile 5 Tabelle 2
Zeile 4 Tabelle 1 Zeile 6 Tabelle 2
Zeile 5 Tabelle 1 Zeile 1 Tabelle 2
Zeile 5 Tabelle 1 Zeile 2 Tabelle 2
Zeile 5 Tabelle 1 Zeile 3 Tabelle 2
Zeile 5 Tabelle 1 Zeile 4 Tabelle 2
Zeile 5 Tabelle 1 Zeile 5 Tabelle 2
Zeile 5 Tabelle 1 Zeile 6 Tabelle 2
Zeile 6 Tabelle 1 Zeile 1 Tabelle 2
Zeile 6 Tabelle 1 Zeile 2 Tabelle 2
Zeile 6 Tabelle 1 Zeile 3 Tabelle 2
Zeile 6 Tabelle 1 Zeile 4 Tabelle 2
Zeile 6 Tabelle 1 Zeile 5 Tabelle 2
Zeile 6 Tabelle 1 Zeile 6 Tabelle 2
```

Wie man sehen kann, multipliziert sich die Menge der in den Tabellen enthaltenen Daten. Das wird im Allgemeinen als **kartesiches Produkt** bezeichnet, und ähnelt dem Vektorprodukt in der Mathematik. Man sollte sich also immer genau Gedanken darüber machen, wie man die Menge der Daten vor einer Weiterverarbeitung (im SQL Statement) einschränken kann.

Ein weiteres Beispiel für eine Beschleunigung um Faktor 100 bei einem einfachen Statement mit **count(*)**. Hiermit sollen die Spalten einer Ausgabe gezählt werden: Versuch 1:

```
select count(*) from tbl where a=# and c=0;

Die Antwortzeiten:

0.16111115813255
0.073595218658447
0.10604506850243
```

MySQL scheint recht schnell zu laufen, allerdings gibt es immer starke Schwankungen bei der Gesamtzeit, bis die Ausgabe auf dem Bildschirm erscheint. Obwohl die internen Zeiten für "time elapsed" recht schnell sind, können durchaus 1 bis 5 Sekunden vergehen, bis das Ergebnis angezeigt wird. MySQL scheint also noch viel Zeit mit anderen Dingen zu verbringen.....

Dasselbe Beispiel ohne **count()**, Versuch 2:

```
select a from tbl where a=# and c=0;

Die Antwortzeiten:

0.0011586213111877
0.0015420305728912
0.0011256754398346
```

tbl besitzt einen eindeutigen index(a,b) und einen nicht eindeutigen index(a,c). Die Zahl der Einträge liegt zwischen 0 und 1000, Versuch 3:

```
select b from tbl where a=# and c=0;

Die Antwortzeiten:

0.0012334552731355
0.0016372671293174
```

```
0.0011582265184521
```

Vergleicht man nun die Ausgabezeiten der Versuche 1-3 miteinander, dann stellt man fest, daß MySQL im dritten Versuch unglaublich schnell geworden ist:

Versuch 1: Durchschnitt ist 0.15165980696678, maximale Ausgabezeit ist 9.3659509420395

Versuch 2: Durchschnitt ist 0.2846958220005, maximale Ausgabezeit ist 5.8125519752502

Versuch 3: Durchschnitt ist 0.2899955201149, maximale Ausgabezeit ist 0.071884989738464

Das Geheimnis liegt in der Anzahl der Verknüpfungen, die der JOIN erzeugt und der Geschwindigkeit der INDEX Tabellen. Es lohnt sich sehr, mit JOINS, INDEXen u.s.w. herumzuspielen. Der Geschwindigkeitsvorteil kann bis zu Faktor 1000 betragen. Folgendes Beispiel macht es vielleicht noch klarer:

Ein weiteres Beispiel mit dem EQUI-JOIN:

```
SELECT A.EineSpalte, B.EineAndereSpalte
FROM Tabelle1 AS A, Tabelle2 AS BWHERE A.Ein Code = B.EinAndererCode;
```

Man kann ihn aber auch ganz anders schreiben, und die Ergebnismenge wird die gleiche sein, nämlich so :

```
SELECT A.EineSpalte, B.EineAndereSpalte
FROM Tabelle1 AS A JOIN Tabelle2 AS B       ON A.Ein Code = B.EinAndererCode;
```

Wenn die Ergebnismenge die gleiche ist, wo liegt dann der Unterschied zwischen diesen beiden Formen ? Gibt es überhaupt einen Unterschied ?

Der Unterschied liegt in der Laufzeit. Im ersten Beispiel wird zuerst das kartesische Produkt aus beiden Tabellen gebildet (jede Zeile aus Tabelle1 wird mit jeder Zeile aus Tabelle2 verknüpft), und wenn beide Tabellen nur jeweils 100 Zeilen enthalten, sind das schon 10.000 Zeilen in der temporären Zwischentabelle. Erst dann werden die Zeilen gelöscht, die nicht die **WHERE** Bedingung erfüllen.

Im zweiten Fall wird zuerst die Bedingung im **ON** Teil geprüft und nur solche Zeilen in die Zwischentabelle übernommen, bei denen die Bedingung erfüllt ist. In dem Beispiel mit den je 100 Zeilen pro Tabelle sind das wahrscheinlich nicht mehr als 100 Zeilen. Das ist ein Faktor von 10 !

Bei noch größeren Tabellen wird dann der Geschwindigkeitsunterschied dramatisch. Faktor 1000 sind dann keine Seltenheit.

Hier noch ein kleines Beispiel:

Ziel: Es soll die Zahl der Personen in einem Ort ausgegeben werden, die keiner Niederlassung angehören:

```
select distinct(a.ort), count(a.id) as anzahl
  from person a,
       rel_niederlassung_person b
 where a.id != b.person_id
 group by ort
 order by ort
```

Tabelle person enthält ca. 4000 records.

Ergebnis: 0.60 sekunden execution time.

Eine andere Möglichkeit ist diese:

```
select distinct(a.ort), count(a.id) as anzahl
     from person a
left join rel_niederlassung_person b
       on a.id != b.person_id
    group by ort
    order by ort
```

Ergebnis: 0.88 Sekunden execution time.

Hier nun eine Möglichkeit, sehen zu können, wie die **MySQL** Datenbank intern arbeitet:

```
EXPLAIN SELECT select_options
```

Wenn einem **SELECT** Statement das Schlüsselwort **EXPLAIN** vorangestellt wird, dann werden Informationen über den Prozeß SELECT ausgegeben, welche Tabellen ausgelesen werden, und in welcher Reihenfolge.

Mit der Hilfe von **EXPLAIN** kann man sehen, wann man **INDIZES** in Tabellen hinzufügen muß, damit die Datenbankabfrage schneller ablaufen kann.

Um eine bestimmte Reihenfolge für JOINS zu erzwingen, muß der Ausdruck STRAIGHT_JOIN angewendet werden.

Ein Beispiel:

```
    EXPLAIN SELECT tt.TicketNumber, tt.TimeIn,
                tt.ProjectReference, tt.EstimatedShipDate,
                tt.ActualShipDate, tt.ClientID,
                tt.ServiceCodes, tt.RepetitiveID,
                tt.CurrentProcess, tt.CurrentDPPerson,
                tt.RecordVolume, tt.DPPrinted, et.COUNTRY,
                et_1.COUNTRY, do.CUSTNAME
            FROM tt, et, et AS et_1, do
            WHERE tt.SubmitTime IS NULL
                AND tt.ActualPC = et.EMPLOYID
                AND tt.AssignedPC = et_1.EMPLOYID
                AND tt.ClientID = do.CUSTNMBR;
```

Ohne daß uns nun die Definitionen der Datenbank interessieren müssen, liefert **MySQL** beispielsweise folgende Ausgabe:

```
    table type possible_keys                key  key_len ref  rows  Extra
    et    ALL  PRIMARY                     NULL NULL    NULL 74
    do    ALL  PRIMARY                     NULL NULL    NULL 2135
    et_1  ALL  PRIMARY                     NULL NULL    NULL 74
    tt    ALL  AssignedPC,ClientID,ActualPC NULL NULL    NULL 3872
          range checked for each record (key map: 35)
```

Die Angabe ALL (type) für alle Tabellen zeigt uns, daß **MySQL** einen vollen JOIN für alle Tabellen durch führen muß. Das wird eine sehr lange Zeit beanspruchen, da hier 74 * 2135 * 74 * 3872 = 45,268,558,720 Reihen durchgearbeitet werden müssen. Also Vorsicht !

Ein Problem mit **MySQL** ist, daß es noch nicht effektiv mit Indizes auf Spalten umgehen kann, falls diese unterschiedlich deklariert sind. In diesem Zusammenhang sind CHAR und VARCHAR identisch, solange diese dieselbe Länge besitzen. Um dieses Problem zu lösen, und MySQL zu beschleunigen muß mit dem Befehl ALTER dafür gesorgt werden, daß die Spalten tt.ActualPC und et.EMPLOYID beide das Format VARCHAR(15) haben:

```
mysql> ALTER TABLE tt MODIFY ActualPC VARCHAR(15);
```

Ein erneutes Starten von EXPLAIN geibt z.B. dann folgende Ausgabe:

```
table type   possible_keys   key     key_len ref         rows    Extra
tt    ALL    AssignedPC,ClientID,ActualPC NULL NULL NULL 3872    where used
do    ALL    PRIMARY         NULL    NULL    NULL        2135
      range checked for each record (key map: 1)
et_1  ALL    PRIMARY         NULL    NULL    NULL        74
      range checked for each record (key map: 1)
et    eq_ref PRIMARY         PRIMARY 15      tt.ActualPC 1
```

Das ist noch nicht perfekt, aber schon um den Faktor 74 schneller !

Eine weitere Optimierung kann dadurch erfolgen, daß die Längen der Vergleiche von tt.AssignedPC = et_1.EMPLOYID und tt.ClientID = do.CUSTNMBR angeglichen werden:

```
mysql> ALTER TABLE tt MODIFY AssignedPC VARCHAR(15),
                      MODIFY ClientID   VARCHAR(15);
```

EXPLAIN zeigt nun:

```
table type   possible_keys   key     key_len ref            rows     Extra
et    ALL    PRIMARY         NULL    NULL    NULL           74
tt    ref    AssignedPC,ClientID,ActualPC ActualPC 15 et.EMPLOYID 52 where
used
et_1  eq_ref PRIMARY         PRIMARY 15      tt.AssignedPC  1
do    eq_ref PRIMARY         PRIMARY 15      tt.ClientID    1
```

Wie man nun sehen kann, haben wir die Datenbank nocheinmal um einen Faktor von 2872 * 2135 = 6.131.720 ! beschleunigt. Besser geht´s nicht. Es ist schon ein Unterschied, ob die Abfrage nur 0.1 Sekunden, oder Stunden dauert !

Viele kommerzielle Datenbanken besitzen sogenannte "query optimizer", in welchen diese automatisch Optimierungen durchführen. In unserem Falle würden wahrscheinlich ORACLE und ADABAS dieses SELECT Statement automatisch verbessern und intern die Abfrage (heimlich) umstrukturieren, sodaß die höchstmögliche Performance auch ohne "Handoptimierung" erzielt würde. Verlassen kann man sich aber nicht darauf. Ich kenne keinen Fall von komplexen, größeren Datenbanken, wo nicht ein Spezialist des Datenbankherstellers für teures Geld vor Ort die Datenbank nocheinmal hätte Optimieren müssen. Vielleicht erinnern Sie sich noch an die Wette von Larry Allison, dem Chef von ORACLE, als er behauptete, ORACLE wäre um den Faktor 1000 bei einem bestimmten Benchmark schneller, als Microsoft SQL Server 6.5 (dahinter steckt SYBASE). Nun, wie wir gesehen haben, kann man also durch intelligente QUERY Optimizer durchaus noch mehr (Faktor Millionen !) an Geschwindigkeit herausholen. Wer sich also im Vorhinein gründlich Gedanken um den Aufbau seiner Datenbank macht, der kann sich Nachoptimierungen ersparen. Man sollte also direkt zu Anfang beim Aufbau der Datenbankstruktur schon einen Spezialisten hinzuziehen, es macht sich schnell bezahlt.

Ein verbleibendes Problem ist, daß **MySQL** annimmt, daß die Werte in der Spalte tt.ActualPC eventuell zerstreut sind, was aber nicht der Fall ist. Es ist aber einfach, dies **MySQL** mitzuteilen:

```
shell> isamchk --analyze Pfad_zur_Mysql_Datenbank/tt
shell> mysqladmin refresh
```

Nun ist die interne Struktur der Tabelle gut geordnet, sodaß ein JOIN optimal ablaufen kann:

```
table type   possible_keys   key     key_len ref            rows    Extra
tt    ALL    AssignedPC,ClientID,ActualPC NULL NULL NULL    3872    where
used
et    eq_ref PRIMARY         PRIMARY 15      tt.ActualPC    1
et_1  eq_ref PRIMARY         PRIMARY 15      tt.AssignedPC  1
do    eq_ref PRIMARY         PRIMARY 15      tt.ClientID    1
```

Der JOIN Optimizer von **MySQL** (ja, MySQL hat auch einen QUERY OPTIMIZER) in der Ausgabe von EXPLAIN hat eine Annahme getroffen (die auch zutrifft), um die Datenbank schneller abzuarbeiten.

Eventuell kann man noch mit dem Operator STRAIGHT_JOIN erreichen, indem man gleichzeitig die Tabellen noch in einer anderen Reichenfolge ordnet...Das hängt dann aber von den Inhalten der Datenbank ab, das ist eher ein statistisches, und kein systematisches Problem. Kommerzielle Datenbanken führen Statistiken über die Inhalte mit, und können somit eventuell die Abfragen noch weiter beschleunigen. Wer also extensiv DATA MINING in riesigen Datenbeständen durchführt, der sollte tunlichst eine gute SQL-Datenbank einsetzen. DATA MINING ist z.B. die Analyse der Unfälle bei einer Versicherung, die versucht, anhand des Autotyps, Fahrleistungen, und sonstigen Angaben das Unfallrisiko und somit die Beitragssätze zu bestimmen. DATA MINING ist sowas wie schwarze Magie, es gibt kaum Experten, die sich damit auskennen, und eventuell noch in der Lage sind, diese Analysen so durchzuführen, daß auch aussagefähige Ergebnisse herauskommen.

Wer also seine Datenbankstruktur testen möchte, der kommt um das Erzeugen von Test-Datensätzen nicht herum.







28.19 DESCRIBE

{DESCRIBE | DESC} tbl_name {col_name | wild}

DESCRIBE gibt Informationen über die Spalten einer Tabelle aus. col_name kann hierbei ein Spaltenname oder ein String sein, der die % oder _ Wildcard Bchstaben enthält.

Falls die Spaltentypen eventuell von MySQL von alleine verändert wurden, dann liegt das an dem eingebauten Optimizer. Siehe hierzu auch Kapitel Heimliche Typenänderungen.

Dieses Statement findet man auch bei ORACLE Datenbanken.

Das Statement SHOW liefert ähnliche Informationen. Siehe hierzu auch Kapitel SHOW.







28.20 LOCK/UNLOCK von TABELLEN

```
LOCK TABLES tbl_name [AS alias] {READ | [LOW_PRIORITY] WRITE}
            [, tbl_name {READ | [LOW_PRIORITY] WRITE} ...]
...
UNLOCK TABLES
```

LOCK TABLES sperrt eine ganze Tabelle für den Thread, also für den Prozeß, der gerade auf die Tabelle zugreift. **UNLOCK TABLES** entsperrt die Tabelle und gibt diese für andere Threads frei. Alle Tabellen, die gesperrt sind, werden automatisch entsperrt, wenn dieser Thread eine andere Tabelle sperrt, oder die Verbindung zum Server löst.

Wenn ein Thread eine READ Sperre setzt, dann kann dieser und alle andere Threads (!) nur aus dieser Tabelle lesen. Also merke: **Ein WRITE Lock sperrt andere Threads für Lese-und Schreibvorgänge, ein READ Lock ermöglicht immer noch das Lesen für alle Threads.**

Jeder Thread wartet solange auf seinen Einsatz, bis er alle angeforderten Locks auch zugewiesen bekommt (was lange dauern kann).

WRITE LOCKS haben normalerweise eine **höhere Priorität** vor **READ LOCKS**.

Damit wird sichergestellt, daß alle Updates so schnell wie möglich eingespielt werden, und die Clients (die eingebenden Personen) nicht mit unnötigen Wartezeiten zu kämpfen haben. Das bedeutet, daß wenn ein Thread einen READ LOCK zugewiesen bekommt, und ein anderer Thread einen WRITE LOCK angefordert, dann werden die Lesezugriffe solange zurückgestellt, bis der Thread den WRITE LOCK erhalten und seine Schreibprozesse beendet hat. WRITE LOCKS kann man aber auch in der Priorität zurücksetzen. Dies geschieht mit Hilfe der Option: LOW_PRIORITY WRITE.

Damit wird der Datenbankserver veranlaßt, nur in Lesepausen die Schreibvorgänge durchzuführen. Dies ist u.a. für Internet Anwendungen interessant. Es muß aber sichergestellt sein, daß es auch tatsächlich Pausen gibt, wo die Schreibvorgänge eingeschoben werden können.

Bei dem Einsatz des Befehls LOCK TABLES muß für alle Tabellen, die in einem Statement abgefragt werden, jeweils ein Lock gesetzt werden. Wenn also eine Tabelle mit mehrfachen ALIAS abgefragt wird, dann muß für jeden ALIAS ein LOCK gesetzt werden. Diese Vorgehensweise dient dazu, die Datenbank von DEADLOCKS freizuhalten.

Man sollte jedoch keinesfalls Tabellen mit einem WRITE LOCK belegen, wenn der Befehl INSER DELAYED verwendet wird. Hierzu muß man wissen, daß **MySQL** den INSERT Befehl einem neuen THREAD zuordnet.

Normalerweise ist ein Lock auf eine Tabelle bei einem einzelnen UPDATE Statement nicht notwendig. Der Grund liegt darin, daß Schreibvorgänge niemals mit Abfragevorgängen kollidieren können, da sie jeweils einen Thread zugewiesen bekommen. Es gibt nur ein paar wenige Fälle, bei denen Tabellen mit einem Lock gesperrt werden müssen, z.B. bei INSERT DELAYED, damit kein SELECT Statement zwischen zwei UPDATE Vorgängen eingeschoben werden kann. Dies würde zu Inkonsistenzen der Tabellen führen.

Merke: Bei der Änderung der Hierarchien von Lese-oder Schreibvorgängen mit ... DELAYED ...., oder durch Änderung der Hierarchie mit Parametern beim Start des `mysqld` Datenbank - Servers, müssen eventuell alle SELECT Statements mit LOCKS versehen werden !

Hier noch ein paar Beispiele zu Problemfällen:

- Falls viele Operationen auf eine ganze Zahl von Tabellen ausgeführt werden müssen, ist es viel schneller, wenn man diese zuvor mit einem Lock sperrt. Natürlich kann dann kein anderer Thread eine Tabelle lesen, die mit einem READ Lock belegt ist, oder in eine Tabelle schreiben, die mit einem WRITE - Lock belegt ist. Trotzdem wird die Datenbank erheblich schneller, da erst dann ein Thread angelegt wird, wenn der entsprechende Lock aufgehoben ist, und CPU Resourcen frei sind. Ohne die Locks würde zuerst ein Thread gestartet, der dann zwangsweise in den Wartezustand versetzt wird, da ja Locks auf der Tabelle liegen. Das kostet bei vielen tausend simultanen Zugriffen erheblich CPU-Zeit, die nirgendwo ausgegeben wird, jedoch deutlich spürbar ist. Ein

Beispiel hierzu findet man im Kapitel Tuning.

- MySQL unterstützt kein Transaktionsmanagement, daher muß eine Tabelle mit einem LOCK gesperrt werden, bevor es nicht mehr passieren kann, daß ein Thread sich zwischen ein **SELECT** und ein **UPDATE** Statement mogelt. Folgendes Beispiel zeigt das Anlegen von LOCKS:

```
mysql> LOCK TABLES trans READ, customer WRITE;
mysql> select sum(value) from trans where customer_id= some_id;
mysql> update customer set total_value=sum_from_previous_statement
           where customer_id=some_id;
mysql> UNLOCK TABLES;
```

- Ohne LOCK gibt es eine hohe Wahrscheinlichkeit, daß sich während der Ausführung des Statements ein anderer Thread zwischengeschoben wird, der z.B. eine neues Tupel (eine neue Zeile) einfügt.
- Dadurch, daß man z.B. **INCREMENTAL UPDATES** durchführt (UPDATE customer SET value=value+new_value) oder die LAST_INSERT_ID() Funktion einsetzt, kann in fast allen Fällen darauf verzichtet werden, die Tabelle mit einem LOCK zu sperren.
- Es gibt allerdings auch **USER LEVEL LOCKs**, die durch **GET_LOCK()** und **RELEASE_LOCK()** gehandelt werden. Diese Locks werden in einer HASH Tabelle im Serverdämon gespeichert, und sind mit den Funktionen **pthread_mutex_lock()** und **pthread_mutex_unlock()** für höchste Geschwindigkeiten ausgelegt.

  Im Kapitel Locks finden Sie genaue Informationen über Lock Eigenschaften.






28.21 SET OPTION

```
SET [OPTION] SQL_VALUE_OPTION= value, ...
```

Der Befehl **SET OPTION** setzt verschiedene Optionen, die sich auf Client und Server auswirken:

```
CHARACTER SET character_set_name | DEFAULT
```

Hier werden alle Character und Strings zwischen Client und Server mit einer Tabelle übersetzt. Momentan ist die einzige einstellbare Option das Character - Set **cp1251_koi8**, allerdings kann dieses einfach geändert werden, in dem man im Quellcode in der Datei **Sql/convert.cc** eine neues Character - Set einstellt, und dann den MySQL Server neu kompiliert. Unter Windows 98/NT ist dies nicht möglich. Um wieder auf das alte Character - Set zurückzustellen, genügt die Angabe der Option **DEFAULT**

PASSWORD = PASSWORD('ein passwort') setzt das Passwort für den momentanen User. Jeder **nicht anonyme** User, der auch offiziell Rechte an seinen Datensätzen besitzt, kann hiermit sein Passwort verändern, um den Zugriff fremder User zu verhindern.

PASSWORD FOR user = PASSWORD('ein passwort') setzt das Passwort für einen bestimmten User auf dem Server. Hierzu ist allerdings nur ein User berechtigt, der Zugang zu der entsprechenden MySQL Datenbank hat. Das Passwort sollte im Format **user@domain.de** angegeben werden, entsprechend den Einträgen in den User und Host Spalten der **mysql.user** Tabelle.

Wenn also in den Spalten für User und Host beispielsweise **Robert** und **%.oracle.de** eingetragen ist, dann muß der Befehl so lauten:

```
mysql> SET PASSWORD FOR bob@"%.loc.gov" = PASSWORD("newpass");
```

SQL_BIG_TABLES = 0 | 1 veranlaßt den MySQL Serverdämon, die Tabellen vorrangig auf der Festplatte zu lagern (Einstellung 1). Bei größeren Tabellen würde ansonsten das komplette RAM und evtl. der Swapspeicher verwendet werden. Andere Dämonen (Apache WWW-Server, CGI-BIN´s) hätten dann evtl. nicht genügend Arbeitsspeicher übrig. Bei der Einstellung 0 werden dann alle Tabellen im RAM gehalten, was zwar etwas schneller ist, allerdings in Versionen < 3.23 zu dem Fehler : **table_name is full** geführt hat. Die Version 3.23 lagert diese HEAP Tabellen selbstständig in ein SWAP-File aus. Der Standardwert bei einer neuen Verbindung zum MySQL Server ist 0. Auf 1 gesetzt, treten größere Performanceeinbrüche bei SELECT Statements in Zusammenhang mit **WHERE** auf. Falls ein SELECT Statement mehr als **max_join_size rows** untersuchen muß, können ebenfalls Performanceeinbrüche auftreten.

SQL_BIG_SELECTS = 0 | 1 auf 1 gesetzt, verhindert, daß MySQL zu lange sucht. Insbesondere bei Statements, deren Laufzeit sich nicht abschätzen läßt, sollte diese Option eingeschaltet werden. Der Standardwert ist hier 0.

SQL_LOW_PRIORITY_UPDATES = 0 | 1 auf 1 gesetzt sorgt dafür, daß **SELECT Abfragen** stets mit **hoher Priorität** vor den Statements **INSERT**, **UPDATE** und **DELETE** abgearbeitet werden. Veränderungen in den Tabellen werden also verzögert durchgeführt. In einigen Fällen müssen, damit Inkonsistenzen vermieden werden, Locks gesetzt werden.

SQL_SELECT_LIMIT = value | DEFAULT gibt eine maximale Zahl von Einträgen an, die durchsucht

werden können. Allerdings kann man mit Hilfe der Angabe von **LIMIT...** diese Grenze überschreiten, damit man z.B. alle Daten seitenweise ausgeben kann. Standardmäßig ist der Wert auf 0 eingestellt, es gibt also keine Begrenzung.

SQL_LOG_OFF = 0 | 1 auf 1 gesetzt bewirkt, daß keine Einträge in die Standard Log-Datei geschrieben werden, falls der Client das Prozess Privileg besitzt. Das Update Log wird davon nicht beeinflußt.

SQL_LOG_UPDATE = 0 | 1 auf 0 gesetzt, verhindert, daß in das Update Log geschrieben wird. Die Standard Log-Datei ist hiervon nicht betroffen !

TIMESTAMP = timestamp_value | DEFAULT setzt die Zeit für den Client. Damit wird sichergestellt, daß die Systemzeit für Client und Server dieselbe ist, insbesondere dann, wenn UPDATE und RESTORE auf Zeilen durchgeführt wird.

LAST_INSERT_ID = # setzt den Wert, der von der Funktion **LAST_INSERT_ID()** zurückgegeben wird. Dieser wird in dem UPDATE Log gespeichert, wenn **LAST_INSERT_ID()** in einem Statement verwendet wird, welches Daten aktualisiert. **INSERT_ID = #** setzt eine Spalte mit AUTO-INCREMENT Eigenschaften auf diesen Wert. Bei **INSERT** und **UPDATE** Statements beginnen die Werte dann nicht mit 1, sondern dem angegebenen Wert. Für verteilte Datenbanken, die eine laufende Zahl als Schlüssel (Key) verwenden, könnte diese Option interessant sein.






28.22 GRANT und REVOKE

GRANT priv_type [(column_list)] [, priv_type [(column_list)] ...] ON {tbl_name | * | *.* | db_name.*} TO user_name [IDENTIFIED BY 'password'] [, user_name [IDENTIFIED BY 'password'] ...] [WITH GRANT OPTION]

REVOKE priv_type [(column_list)] [, priv_type [(column_list)] ...] ON {tbl_name | * | *.* | db_name.*} FROM user_name [, user_name ...]

GRANT ist erst seit der Version MySQL 3.22.11 implementiert. MySQL hat ein fortschrittliches, aber vom Standard abweichendes Sicherheits/Rechte-System. Was kann das Rechte-System erledigen ?

Die grundlegende Funktion des MySQL-Rechte-Systems ist es, einem Usernamen auf einem Host die SELECT, INSERT, UPDATE und DELETE Rechte bzgl. einer Datenbank zu erteilen. Außerdem besteht die Möglichkeit, einem "anonymous"-User bestimmte Dinge zu erlauben, und eventuell auch Statements, wie z.B. LOAD DATA INFILE auszuführen. In der neuesten Version 3.23 ist auch die Vergabe von Rechten in Abhängigkeit der Hostadresse, der IP-Nummer oder der Netzwerk-Adresse möglich. Host Nummer und Userpasswort sind dabei unabhängig voneinander, was bedeutet, daß es z.B. zwei User mit demselben Usernamen geben darf, vorausgesetzt, daß diese sich stets aus anderen Netzwerken einloggen. Es sollte beachtet werden, daß die Usernamen für die MySQL - Datenbank nichts mit UNIX - Usern oder Microsoft Windows NT/98 Konten zu tun haben.

MySQL versteht die **Kombination** aus einem **Hostname**, einer **Netzwerkadresse** oder einer **IP-Nummer** in Verbindung mit einem einem **User** als eindeutige Identität. Aufgrund dieser Identität entscheidet das GRANT System dann welche Rechte ein Client erhält. Die Rechte können jederzeit mit dem Skript **mysqlaccess** getestet werden.

Alle Rechte werden in **drei** Tabellen: **user**, **host** und **db** gespeichert:

Tabellen Name	user	db	host
Scope Felder	Host	Host	Host
	User	Db	Db
	Password	User	
Privileg Felder	Select_priv	Select_priv	Select_priv
	Insert_priv	Insert_priv	Insert_priv
	Update_priv	Update_priv	Update_priv
	Delete_priv	Delete_priv	Delete_priv
	Index_priv	Index_priv	Index_priv
	Alter_priv	Alter_priv	Alter_priv
	Create_priv	Create_priv	Create_priv
	Drop_priv	Drop_priv	Drop_priv
	Grant_priv	Grant_priv	Grant_priv
	Reload_priv		
	Shutdown_priv		
	Process_priv		
	File_priv		

Jedes in der User-Tabelle erteilte Recht ist für alle Datenbanken gütig, die in der db-Tabelle nicht gefunden werden können. Deshalb sollte man einzelnen Usern (vom Supervisor abgesehen) Rechte nur auf Datenbank-Ebene zuteilen.

Die Host-Tabelle existiert hauptsächlich, um eine Liste "sicherer" Server zu verwalten. Bei TcX enthält die Host-Tabelle eine Liste aller Rechner im lokalen Netzwerk.

Die Rechte des sich gerade anmeldenden Users werden nach dem folgenden Algorithmus festgestellt:

Die Tabelle host wird nach Hosts ohne Wildcard, gefolgt von hosts mit einer Wildcard und Einträgen mit host=""

sortiert. Innerhalb jedes hosts, wird nach den gleichen Regeln nach dem user sortiert. Die Tabelle db wird nach den gleichen Regeln sortiert. Bei den nachfolgenden Schritten wird in den so sortierten Datensätzen nachgesehen und der erste passende Datensatz verwendet.

Die Rechte des sich anmeldenden Benutzers werden aus der Tabelle user entnommen. Dabei wird, wie bereits weiter oben beschrieben, der erste passende Datensatz aus der vorher sortierten Tabelle verwendet. Den so erhaltenen Satz an Rechten nennen wir einmal **PRIV**. Die Rechte des sich anmeldenden Benutzers werden aus der Tabelle db entnommen. Auch hier wird die vorher sortierte Tabelle, und der erste passende Datensatz verwendet.

Falls der in der db-Tabelle gefundene Datensatz den Eintrag host="" enthält, werden die ursprünglichen Rechte **PRIV** aus der user-Tabelle mit den Host-Rechten aus der host-Tabelle logisch **UND** verknüpft. D.h. im Klartext: Aus beiden Datensätzen werden alle Rechte entfernt. bei denen nicht in in beiden Fällen ein "Y" eingetragen ist. Falls host

"" ist , so werden die Rechte von PIV nicht verändert. In solchen Fällen muß der host-Eintrag zumindest teilweise mit dem Hostname des verbindenden Hosts übereinstimmen. Deshalb kann angenommen werden, daß die in dieser Zeile festgelegten Rechte dem Profil des sich anmeldenden hosts entsprechen.

Die Rechte des Users aus der Tabelle **user** werden anschließend mit dem PRIV Rechtesatz logisch **ODER** verknüpft. (d.h. alle Y-Rechte werden hinzugefügt).

Achtung: Falls in den Rechte-Tabellen etwas verändert wurde, muß das Kommando: **mysqladmin reload** durchgeführt werden, damit die Änderungen aktiv werden.

Der sich einloggende User erhält dann die PRIV Rechte zugewiesen. Nachfolgend ein Beispiel für das Sortieren und Auffinden der richtigen Datensätze. Angenommen, die user-Tabelle sieht folgendermaßen aus:

```
+-----------+---------+-
| Host      | User    | ...
+-----------+---------+-
| %         | root    | ...
| %         | jeffrey | ...
| localhost | root    | ...
| localhost |         | ...
+-----------+---------+-
```

Die Suchreihefolge sieht dann (nach erfolgter Sortierung) folgendermaßen aus:

```
localhost/root
localhost/any
any/jeffrey
any/root
```

Jeffrey der sich via **localhost** anmeldet (also von der Arbeitsstation, auf der **mysqld** läuft), und wird deshalb mit den **Rechten localhost/any** und **nicht** mit den Rechten **any/jeffrey** ausgestattet, da immer der erste passende Eintrag verwendet wird !

Falls Sie also Probleme mit den Zugriffsrechten haben, lassen Sie sich den Inhalt der Tabelle user ausgeben, sortieren ihn von Hand und stellen den ersten passenden Datensatz fest.

Es folgt nun ein Beispiel um den user "custom" hinzuzufügen der sich von den Hosts "localhost", "server.domain" und "whitehouse.gov" anmelden darf. Er möchte das Password "stupid" haben. Die Datenbank "bankaccount" möchte er nur via "localhost", die Datenbank "customer" von allen drei hosts aus erreichen können:

```
shell> mysql mysql.
mysql> insert into user (host,user,password)
       values('localhost','custom',password('stupid'));
mysql> insert into user (host,user,password)
       values('server.domain','custom',password('stupid'));
```

```
mysql> insert into user (host,user,password)
       values('whitehouse.gov','custom',password('stupid'));

mysql> insert into db
       (host,db,user,Select_priv,Insert_priv,Update_priv,Delete_priv,
        Create_priv,Drop_priv)
       values
       ('localhost','bankaccount','custom','Y','Y','Y','Y','Y','Y');
mysql> insert into db
       (host,db,user,Select_priv,Insert_priv,Update_priv,Delete_priv,
        Create_priv,Drop_priv)
       values
       ('%','customers','custom','Y','Y','Y','Y','Y','Y');
```

Sie können selbstverständlich xmysqladmin, mysql_webadmin, mysqladmin und xmysql oder PHPmyAdmin verwenden, um in den Rechte-Tabellen Datensätze einzufügen oder abzuändern. Sie finden diese Utilities u.a. im Support Verzeichnis unter http://www.rent-a-database.de/support/.

Die Rechtevergabe-Tabellen

Name	tables_priv	columns_priv
Scope Felder	Host	Host
	Db	Db
	User	User
	Table_name	Table_name
		Column_name
Privileg Rechte	Table_priv	Column_priv
	Column_priv	
Andere Rechte	Timestamp	Timestamp
	Grantor	

Der "SCOPE" ist immer der Wirkungsbereich, daher werden in der Rechtevergabe-Tabelle genau festgelegt, auf welche Felder ein User zugreifen darf.

Die für Tabellen relevanten Rechte sind select, insert, update und delete. Die Rechte für Tabellen und Datenbanken sind create und drop. "create" und "drop"-Recht gelten für beides - Tabellen und Datenbanken.

Falls ein User nämlich alle Tabellen einer Datenbank löschen darf, kann er auch gleich das Recht erhalten, die ganze Datenbank zu löschen.

Die Zeile **Andere Rechte** vergeben die Berechtigung um Dateien zu benutzen (für LOAD DATA INFILE und SELECT INTO OUTFILE) und Administrations-Kommandos wie shutdown, reload, refresh und process auszuführen.

Das Rechtevergabesystem basiert auf 3 Tabellen:

user Tabelle

Sie beinhaltet alle host+user Kombinationen, die das Recht haben, sich an dem MySQL - Server anzumelden, und enthält gegebenenfalls noch deren Paßwort. Die user-Tabelle hat die folgenden Datenfelder:

Datenfeld	Datentyp	Schlüssel	Standardwert
Host	char(60)	PRI	""
User	char(16)	PRI	""
Password	char(16)	-	""
Select_priv	enum('N','Y')	-	N
Insert_priv	enum('N','Y')	-	N
Update_priv	enum('N','Y')	-	N
Delete_priv	enum('N','Y')	-	N

```
Create_priv     enum('N','Y')   -               N
Drop_priv       enum('N','Y')   -               N
Reload_priv     enum('N','Y')   -               N
Shutdown_priv   enum('N','Y')   -               N
Process_priv    enum('N','Y')   -               N
File_priv       enum('N','Y')   -               N
```

Die **db-Tabelle** legt fest, welche Datenbanken **host+user** benutzen kann, und was er mit den Tabellen in jeder Datenbank machen darf. Die db-Tabelle hat die folgenden Datenfelder:

```
Datenfeld       Typ             Schlüssel       Standardwert
Host            char(60)        PRI             ""
Db              char(64)        PRI             ""
User            char(16)        PRI             ""
Select_priv     enum('N','Y')   -               N
Insert_priv     enum('N','Y')   -               N
Update_priv     enum('N','Y')   -               N
Delete_priv     enum('N','Y')   -               N
Create_priv     enum('N','Y')   -               N
Drop_priv       enum('N','Y')   -               N
```

Die **host-Tabelle** wird nur in großen Netzwerken benutzt, um bei leeren host-Einträgen in der db-Tabelle nachzuschauen. D.h. falls sie einem User gestatten wollen, Datenbanken von jedem Host in ihrem Netzwerk zu benutzen, sollten Sie "" als Hostname in der db-Tabelle eintragen. In diesem Fall sollte die host-Tabelle für jeden Host ihres Netzwerks einen Eintrag aufweisen. Die host-Tabelle hat die folgenden Datenfelder:

```
Datenfeld       Typ             Schlüssel       Standardwert
Host            char(60)        PRI             ""
Db              char(64)        PRI             ""
Select_priv     enum('N','Y')   -               N
Insert_priv     enum('N','Y')   -               N
Update_priv     enum('N','Y')   -               N
Delete_priv     enum('N','Y')   -               N
Create_priv     enum('N','Y')   -               N
Drop_priv       enum('N','Y')   -               N
```

Die host und db Spalten können eine Zeichenkette mit der SQL-Wildcard % und _ enthalten. Wenn irgendeine dieser Spalten leer gelassen wird, so ist das gleichbedeutend mit dem Eintrag '%'.

Ein host kann localhost, ein Hostname, eine IP-Nummer, eine Netzwerknummer oder eine Zeichenkette mit Wildcards sein. Ein leerer Host in der db-Tabelle ist gleichbedeutend mit **alle Hosts** in der host-Tabelle. Ein leerer Host in der host- oder user-Tabelle besagt, daß alle Hosts eine TCP-Verbindung zu ihrem MySQL-Server herstellen können.

db ist der Name einer Datenbank oder ein SQL regulärer Ausdurck (regexp). Eine leere user Spalte bedeutet, daß jeder Username gültig ist. Wildcards im Feld Username sind nicht zulässig.

Ein user, auf den nichts in der user-Tabelle paßt, wird wie ein no-name user oder anonymous User behandelt.

Die Rechte der user-Tabelle werden mit der db-Tabelle über **ODER** verknüpft. Das bedeutet, daß der Superuser einfach nur einen Eintrag in der user-Tabelle haben muß, in dem alle Rechte-Flags auf Y gesetzt sind.

Sie können einen Eintrag wie 123.444.444.% in der host-Tabelle verwenden um jedem User eines IP-Klasse-C-Netzes Zugriff zu ermöglichen.

Um zu verhindern, daß jemand durch Benennung seines Hosts als 123.444.444.somewhere.com in das System eindringt, verbietet MySQL alle Hostnamen, die mit Zahlen oder einem Punkt beginnen.

Falls also der Rechner 1.2.foo.com oder ähnlich heißt, wird er durch Namensüberprüfung keinen Zugang erhalten.

Verwenden sie in diesem Fall seine IP-Nummer in der Rechtevergabe.

Hinzufügen neuer User-Rechte zu MySQL

Dies setzt voraus, daß der gegenwärtige User **INSERT-Rechte** für die mysql Datenbank und auch die RELOAD-Rechte besitzt. Der DB-Server (mysqld) muß gestartet sein. Falls er noch nicht gestartet ist, starten sie ihn mit dem Befehl **safe_mysqld --log &**:

```
> mysql mysql
  insert into user values
('%','monty',password('something'),'Y','Y','Y','Y','Y','Y','Y','Y','Y','Y')
;
  insert into user (host,user,password) values('localhost','dummy',") ;
  insert into user values
('%','admin',",'N','N','N','N','N','N','Y','N','Y','Y') ;
  quit
> mysqladmin reload
```

Dies erzeugt drei neue User:

Monty - Voller superuser, muß aber ständig ein Passwort bei der Arbeit mit MySQL benutzen.

admin - Braucht kein Passwort, kann aber nur mysqladmin reload, mysqladmin refresh und mysqladmin processlist ausführen. Es können ihm individuelle Datenbank-Rechte durch die db-Tabelle eingeräumt werden.

dummy - Ihm müssen individuelle Datenbank-Rechte durch die db-Tabelle eingerichtet werden.

Default Rechte

Die default Rechte (wurden durch "Scripts/mysql_install_db" gesetzt) erlauben dem User root alles. Jeder User kann mit jeder Datenbank, deren Name 'test' ist oder mit 'test_' beginnt, alles tun. Ein normaler User kann mysqladmin shutdown oder mysqladmin processlist nicht benutzen. Sehen Sie sich das Skript (Scripts/mysql_install_db') als Beispiel an, und sehen Sie, wie man weitere User hinzufügt. Die Rechte-Tabellen werden mit **mysqladmin** reload eingelesen. MySQL benutzt also einfache Tabellenstrukturen, um die Rechte zu vergeben, eine naheliegende Variante.

Ein Beispiel für die Rechte-Vergabe

Ein üblicher Fehler ist es, zu vergessen, daß Paßworte verschlüsselt abgespeichert werden. Dies führt zum Beispiel zu folgendem Problem:

```
INSERT INTO user VALUES
('%','jeffrey','bLa81m0','Y','Y','Y','N','N','N','N','N', 'N','N');
```

Danach muß **mysqladmin reload** ausgeführt werden, um die Änderung zu aktivieren. Es folgt ein Versuch sich beim Server anzumelden:

```
$ ./mysql -h sqlserver -u jeffrey -p bLa81m0 test
Access denied
```

Versuchen sie stattdessen folgendes:

```
INSERT INTO user VALUES
('%','jeffrey',password('bLa81m0'),'Y','Y','Y','N','N','N','N','N','N','N');
```

Nun sollte es wie gewünscht funktionieren.

Lösungen bei Access denied - Fehlern

Haben sie die MySQL-Rechte-Tabellen mit dem Skript **mysql_install_db** installiert? Testen sie dies, indem Sie den Befehl **mysql -u root test** ausführen. Hier dürfte kein Fehler erscheinen.

Bei einer Erstinstallation sollten sie 'mysql -u root mysql' benutzen, um Zugang zu den Rechte-Tabellen zu erhalten.

Für Testzwecke sollten sie den mysqld - Dämon mit der Option **--without-grant-tables** starten. Nun können sie die MySQL-Rechte-Tabellen ändern und mit Hilfe des Skripts mysqlaccess überprüfen, ob ihre Einstellungen funktionieren. mysqladmin reload sagt dann dem mysqld-Dämon, daß er die neuen Rechte-Tabellen benutzen soll.

Selbst wenn sie mit Perl, Python, oder ODBC Zugriffsprobleme haben, sollten sie immer mit **mysql -u user database** oder **mysql -u user -password database** ihre Probleme mit Rechten testen.

Man kann auch die Option **--password=your_password** beim Start von MySQLD verwenden, um das Passwort zu übergeben.

Falls sie bei einer Anmeldung mit **mysql -u user database** die Fehlermeldung 'Access denied' erhalten, dann haben sie ein Problem mit der 'user'-Tabelle.

Überprüfen sie dies durch **mysql -u root mysql** und **select * from user**. Sie sollten einen Eintrag mit 'hostname' und 'user' erhalten der auf ihren Rechnername und ihren Usernamen paßt.

Die Fehlermeldung "Access denied" sagt ihnen, wer sie sind, von welchem Host aus sie sich anmelden und ob sie ein Passwort benutzt haben oder nicht. Sie sollten normalerweise einen Eintrag in der user-Tabelle haben, der genau wie in der Fehlermeldung angegeben, auf ihren Rechner- und Username paßt.

Falls sie die Fehlermeldung 'Host ... is not allowed to connect to this MySQL server' erhalten, wenn sie versuchen sich an einem MySQL-Server auf einer anderen Maschine anmelden, dann haben sie keinen Datensatz in der User-Tabelle, die auf die Remote-Maschine paßt. Sie können dies beheben, indem sie das Kommandozeilen-Tool 'mysql' benutzen, und für die Maschinen/User-Kombination mit der sie sich anmelden wollen, einen Datensatz zur user-Tabelle hinzufügen. Falls sie nicht MySQL 3.22 installiert haben, und nicht genau wissen, mit welcher IP/Hostname-Kombination sie sich anzumelden versuchen, dann sollten sie einen Eintrag mit '%' als Host in der user-Tabelle einfügen und mysqld mit der Option **--log** neu starten. Nach einem Anmeldeversuch finden sie dann Informationen im MySQL Log, wie sie sich tatsächlich angemeldet haben.

Falls **mysql -u root test** funktioniert, aber **mysql -h your_hostname -u root test** die Fehlermeldung 'Access denied' erzeugt, dann haben sie nicht den richtigen Namen für ihren Rechner in der user-Tabelle.

Falls sie z.B. einen Eintrag mit host 'tcx' in der 'user'-Tabelle haben, ihr DNS aber dem MySQL-Server sagt, daß ihr Hostname 'tcx.subnet.se' ist, dann wird dieser Eintrag nicht funktionieren. Probieren sie es mit einem Datensatz, der die IP-Nummer ihres Hosts in der 'user'-Tabelle enthält.

Sie können natürlich auch einen Host mit einer Wildcard (z.B. 'tcx%') zur 'user'-Tabelle hinzufügen. Beachten sie aber bitte, daß Hostnamen die mit '%' aufhören, eine Sicherheitslücke darstellen !

Falls es ihnen nicht gelingt, festzustellen, warum sie die Fehlermeldung 'Access denied' erhalten, dann entfernen sie alle Einträge, die Wildcards (Einträge die % oder _ enthalten) im Hostnamen enthalten, aus der user-Tabelle ! Ein häufiger Fehler ist es, daß man einen neuen User hinzufügt der host='%' und user='some user' hat und glaubt, daß man sich auch vom demselben Rechner, also **localhost** anmelden darf.

Der Grund, warum das nicht funktioniert, ist, daß der Eintrag mit host='localhost' und **user=""** gegenüber dem neuen Eintrag bevorzugt wird, wenn man sich von 'localhost' anmeldet. Die richtige Vorgehensweise ist es dann, eine zweiten Datensatz mit host='localhost' und user='some_user' vorzunehmen, oder den Eintrag mit **user=""** zu entfernen.

Bei der Benutztung der MIT-pthreads, wird localhost nie benutzt. Alle Verbindungen zum mysql-Daemon laufen über TCP/IP und sie müssen ihren richtigen Hostnamen in 'user' eingetragen haben, selbst wenn sie den Client auf dem gleichen Rechner benutzen, auf dem auch der Server läuft.

Falls sie die Fehlermeldung 'Access to database denied' erhalten, dann haben sie ein Problem mit der db-Tabelle. Falls der benutzte Eintrag in der db-Tabelle einen leeren Hostnamen hat, dann überprüfen sie auch den zughörigen Eintrag in der 'host'-Tabelle.

Falls **mysql -u user database** auf dem Server-Rechner funktionert, **mysql -u host -u user database** aber nicht auf einem anderen Client-Host funktioniert, dann haben sie diesen Client-Rechner nicht in der 'user'- oder 'db'-Tabelle eingetragen.

Falls das Paßwort abgelehnt wird, dann denken sie daran, daß Paßworte mit der PASSWORD-Funktion eingefügt werden müssen. Schauen sie im Abschnitt Ein Beispiel für die Rechte-Vergabe nach.

Falls **mysql -u user test** funktioniert, **mysql -u user other_database** hingegen nicht, dann ist die **other_database** nicht in der 'db'-Tabelle aufgeführt.

Falls Sie die Fehlermeldung 'Access to database denied' bei der Benutzung von **SELECT ... INTO OUTFILE** oder **LOAD DATA INFILE** Befehlen erhalten, dann ist wahrscheinlich das **File_priv** - Recht für Sie selber nicht in der 'user'-Tabelle aufgeführt.

Falls alles fehlschlägt, dann starten sie den mysqld Daemon mit: --debug=d,general,query. Dadurch werden Informationen über den Host und User ausgegeben, der sich anzumelden versucht, und außerdem erhalten sie Informationen über jedes Kommando.

Lesen Sie hierzu auch das Kapitel Debuggen von MySQL

Falls sie irgendwelche weiteren Probleme mit den MySQL-Rechte-Tabellen, und das Gefühl haben, diese auf der Mailing-Liste zu veröffentlichen, dann fügen sie immer einen Dump der MySQL-Rechte-Tabellen hinzu. Sie können diesen Tabellendump mit dem Befehl mysqldump mysql durchführen. Wie immer, mailen sie ihr Problem mit dem mysqlbug-Skript.

Falls sie die Fehlermeldung **Can't connect to local mySQL server** oder **Can't connect to MySQL server on some_hostname erhalten**, so bedeutet dies, daß der mysqld-Daemon nicht gestartet ist, oder daß sie versuchen, sich an dem falschen Port anzumelden. Stellen sie sicher, daß der MYSQLD auf dem richtigen Port läuft:

```
telnet hostname 3306
```

Alternativ sollten Sie bei neueren Versionen den Port 3333 ausprobieren.

Unbedingt sollte Sie sich das Kapitel Wie sichere ich MySQL gegen Hacker ab ? durchlesen.






28.23 CREATE INDEX

```
CREATE [UNIQUE] INDEX index_name ON tbl_name (col_name[(length)],... )
```

Das CREATE INDEX Statement funktioniert erst seit der Version 3.22. Zu Funktionsweise schauen Sie bitte im Kapitel ALTER TABLE nach.

Normalerweise werden automatisch Indizes auf allen Tabellen angelegt, die mit dem CREATE TABLE Statement angelegt worden sind. CREATE INDEX erlaubt es nun, einen INDEX zu existierenden Tabellen hinzuzufügen.

Wenn eine Spaltenliste der Form (spalte1, spalte2,...) angegeben wird, wird ein mehrspaltiger INDEX erzeugt. Dies ist immer dann sinnvoll, wenn man einen eindeutigen Schlüssel (Primärschlüssel) nur aus der Kombination mehrerer Einträge erhält (Es macht keinen Sinn, einen INDEX allein auf den Nachnamen zu legen (Schmitz gibt´s zu viele), sondern eindeutig wird eine Person erst durch die Zuordnung mehrerer Charakteristika (Nachname und Geburtsdatum und Wohnort).

Für CHAR und VARCHAR Spalten werden stets nur die ersten 10 Buchstaben indiziert. Beispiel:

```
mysql> CREATE INDEX part_of_name ON customer (name(10));
```

Da die meisten Namen sich innerhalb der ersten 10 Buchstaben unterscheiden, sollte der INDEX nicht kleiner sein, als der INDEX für die ganze Namensspalte.

In Kapitel Wie MySQL Indexe verwendet finden Sie weitere Informationen zu Indizes.







28.24 DROP INDEX

DROP INDEX index_name

Die Funktion DROP INDEX funktioniert erst ab der Version 3.22. Siehe hierzu auch ALTER TABLE.







28.25 Comment

Der MySQL Server ignoriert nach C und PERL Syntax Kommentare. Alles, was in den Statements **/* ....*/** eingeschlossen ist, oder nach dem # Zeichen in einer Zeile folgt, wird als Kommentar ignoriert.

```
mysql> select 1+1;     # This comment continues to the end of line
mysql> select 1 /* this is an in-line comment */ + 1;
mysql> select 1+
/*
this is a
multiple-line comment
*/
1;
```







28.26 CREATE FUNCTION/DROP FUNCTION

```
CREATE FUNCTION function_name RETURNS {STRING|REAL|INTEGER}
       SONAME shared_library_name

DROP FUNCTION function_name
```

Die User definierbaren Funktionen, auch UDF Funktionen genannt, sind ein Weg, MySQL mit einer neuen Funktion auszustatten, die so, wie z.B. ABS() oder CONCAT() funktioniert.

CREATE FUNCTION speichert den Namen der Funktion, Typ und die Shared Library in der **mysql.func** Systemtabelle. Der User muß sowohl **insert** als auch **delete** Privilegien besitzen, um Funktionen zu schaffen, oder diese zu löschen.

Alle aktiven Funktionen werden jedesmal neu geladen, sobald der Server neu startet. Eine Ausnahme ist es, wenn der **mysqld** Dämon mit der Option --skip-grant-tables gestartet wird. Dann sind UDF Funktionen nicht verfügbar. Damit UDF funktioniert, müssen diese in C++ geschrieben werden, und das dynamischen Laden von Libraries muß möglich sein. MySQL darf nicht mit der Option -static kompiliert sein.




28.27 Ist MySQL empfindlich bei reservierten Worten ?

Der Parser bei MySQL kann die Syntax nur dann korrekt erkennen, wenn bestimmte Ausdrücke für SQL reserviert bleiben:

```
action          add             all             alter           after
and             as              asc             auto_increment  between
bigint          bit             binary          blob            bool
both            by              cascade         char            character
change          check           column          columns         constraint
create          cross           current_date    current_time    current_timestamp
data            database        databases       date            datetime
day             day_hour        day_minute      day_second      dayofmonth
dayofweek       dayofyear       dec             decimal         default
delete          desc            describe        distinct        distinctrow
double          drop            escaped         enclosed        enum
explain         exists          fields          first           float
float4          float8          foreign         from            for
full            function        grant           group           having
hour            hour_minute     hour_second     ignore          in
index           infile          insert          int             integer
interval        int1            int2            int3            int4
int8            into            join            key             keys
last_insert_id  leading         left            like            lines
limit           load            lock            long            longblob
longtext        low_priority    match           mediumblob      mediumtext
mediumint       middleint       minute          minute_second   month
monthname       natural         numeric         not             null
on              option          optionally      or              order
outer           outfile         partial         password        precision
primary         procedure       processlist     privileges      quarter
read            real            references      rename          regexp
reverse         repeat          replace         restrict        returns
rlike           second          select          set             show
smallint        soname          sql_big_tables  sql_big_selects sql_select_limit
sql_low_priority_updates        sql_log_off     sql_log_update  straight_join
starting        status          string          table           tables
terminated      text            time            timestamp       tinyblob
tinytext        tinyint         trailing        to              use
using           unique          unlock          unsigned        update
usage           values          varchar         variables       varying
varbinary       with            write           where           year
year_month      zerofill
```

Einige Ausdrücke werden sehr gerne verwendet, hier also noch eine Auswahl aus den obigen Ausdrücken, die nicht verwendet werden dürfen:

```
ACTION
BIT
DATE
ENUM
NO
TEXT
TIME
TIMESTAMP
```







29. Wie sichere ich MySQL gegen Hacker ab ?

Um ein MySQL System abzusichern, sollten man folgendes bedenken:

Benutzen Sie Passörter für alle MySQL Benutzer. Denken sie daran, daß jeder sich als eine andere Person anmelden kann, indem er z.B. 'mysql --user other_user database' verwendet. Das ist übliches Verhalten bei allen Client/Server-Applikationen. Sie können das Passwort aller User ändern indem sie das mysql_install_db-Skript editieren, oder wenn es nur um den MySQL root user geht durch:

```
mysql -u root -e "update user set password=password('new_password')
where user='root'" mysql
```

Starten sie den MySQL Dämon nicht als root. mysqld kann von jedem Useraccount des Betriebssystems gestartet werden.

Sie können auch einen neuen User 'mysql' anlegen um alles noch sicherer zu machen. Sie brauchen den root user name in der 'user'-Tabelle nicht zu ändern, selbst wenn sie mysqld unter einem anderen User starten, da der Username, unter dem der MySQLD - Dämon gestartet wird, nichts mit den Userrechten des GRANT Mechanismus zu tun hat.

Um **mysqld** mit anderen User-Rechten zu starten, genügt es, das mysql.server Skript zu editieren. Dies wird normalerweise mit dem 'su'-Kommando ausgeführt.

Stellen sie sicher, daß nur der mysqld - User die read/write Rechte auf die Datenbank-Verzeichnisse hat.

Geben sie nicht allen Usern die 'Process_priv' Rechte. Jeder kann mit dem 'mysqladmin processlist'-Kommando überprüfen, ob jemand das Password ändert.

mysqld speichert automatisch besondere Rechte für User mit 'Process_priv'-Rechten in der User Tabelle ab, so daß ein root user sich einloggen kann, selbst wenn alle normalen Verbindungen in Benutzung sind.

Geben sie 'File_priv' - Rechte nicht an alle User. Wenn ein User dieses Recht hat, kann er eine im ganzen Dateisystem Dateien mit den Rechten des mysql-Daemon anlegen !

Um dies zu verhindern, werden alle Dateien, die mit SELECT INTO OUTFILE erzeugt werden, mit dem Leserecht für jedermann versehen, damit es nicht möglich ist, bereits bestehende Dateien zu überschreiben.

Falls sie ihrem DNS Server nicht trauen, sollten sie IP-Nummern anstatt Hostnamen in den Rechte-Tabellen verwenden.

Die Option **--secure** für mysqld sollte im Prinzip Hostnames mit einem **double reverse lookup** absichern.

Auf jeden Fall sollte man sehr vorsichtig mit Wildcards in Hostnamen sein !

Falls sie ein root Paßwort in das mysql.server Skript integrieren, dann sollten sie dieses nur für root lesbar machen.

Die nachfolgenden Optionen beinflußen die mysqld Sicherheit:

--secure

Stelle sicher, daß die IP-Nummer, die von **get_hostbyname()** geliefert wurde, sich auch auf den Original-Hostnamen zurück abbilden läßt. Dies wurde implementiert, um es für jemand außerhalb schwieriger zu machen, sich Zugang zu verschaffen, indem er einen anderen Host vortäuscht (adress spoofing).

Diese Option fügt auch einige "sanity checks" für Hostnamen hinzu. Dies ist bei MySQL 3.21 defaultmäßig abgeschaltet, da es manchmal lange dauert dies zu überprüfen. Der User hat dann fälschlicherweise den Eindruck, MySQL wäre zu langsam ...

MySQL ab Version 3.22 speichert Hostnamen zwischen und hat dies defaultmäßig eingeschaltet.

--skip-grant-tables

Benutze das Rechte-System nicht. Dies gibt jedermann vollen Zugriff auf alle Datenbanken !

--skip-name-resolve

Hostnamen werden nicht aufgelößt. Alle Hostnamen in den Rechte-Tabellen müssen IP-Nummern oder 'localhost' sein.

--skip-networking

Erlaube keine Verbindungen über das Netzwerk (TCP/IP). Alle Verbindungen zu mysqld werden mit Unix-Sockets durchgeführt. Diese Option funktioniert auf Systemen mit MIT-pthreads nicht sehr gut, da MIT-pthreads die Unix-Sockets nicht unterstützen.

Hier noch einmal die Privilegien im Überblick:

```
ALL PRIVILEGES      FILE                RELOAD
ALTER               INDEX               SELECT
CREATE              INSERT              SHUTDOWN
DELETE              PROCESS             UPDATE
DROP                REFERENCES          USAGE
```

ALL ist ein Synonym für ALL PRIVILEGES. REFERENCES ist leider noch nicht implementiert.

USAGE ist momentan (könnte noch geändert werden) ein Synonym für "no privileges". Es kann für einen User verwendet werden, der keine Privilegien besitzt.

Um Rechte einem User zu entziehen, muß man den Wert mit der **GRANT** Option verändern.

```
REVOKE GRANT OPTION ON ... FROM ...;
```

Die einzigen **priv_type** Werte, die man für einen tabelle verändern kann, sind **SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, GRANT, INDEX** und **ALTER**.

Es können auch **priv_type** Werte für Spalten angegeben werden. Diese sind **SELECT, INSERT** and **UPDATE**.

Man kann globale Privilegien mit Hilfe der **ON *.*** Syntax setzen, wie z.B.: **ON db_name.* ....**. Aber Achtung ! Wer keine Datenbank angegeben hat, setzt die Privilegien für die komplette Datenbank.

Für bestimmte User, die immer über dieselbe IP-Nummer sich an den Server anbinden, kann es interessant sein, in MySQL dieses mitzuloggen und zu regeln. Hierzu kann man entweder einen User mit der Form

user@host oder auch mit Wildcards bestimmen, wie z.B. mit dem Zeichen "%".

So könnte man auch **user@"%.domain.de"**, oder **user@"192.168.100.%"** angeben, um einem User in einem Klass-C Netz Zugang zu geben. Dieses Beispiel sollte immer dann verwendet werden, wenn die IP-Nummern im Netz mit DHCP vergeben werden.

Der einfache User ist ein Synonym für user@"%". Wenn jedoch der Zugang für alle User freigegeben wird, dies ist die Default - Einstellung, dann sollten Sie unbedingt alle lokalen User noch in die **mysql.user** (Datenbank mysql, Tabelle **user** Tabelle eintragen. Es werden dann alle User Ihres Netzwerkes korrekt mitgeloggt.

So können Sie später ermitteln, welcher Ihnen bekannte User auf die Datenbank zugegriffen hat:

```
mysql> SELECT Host,User FROM mysql.user WHERE User=";
```

Im Moment unterstützt GRANT nur **host, table, database** und **columns** Rechte mit bis zu 60 Buchstaben. Ein Username darf maximal 16 Buchstaben lang sein, was in der Praxis völlig ausreichen dürfte.

Die Privilegien für eine Tabelle oder eine Spalte werden in einem logischen OR mit anderen Privilegien in anderen Tabellen verknüpft. Entweder Sie sind als User berechtigt, auf die Datenbank zuzugreifen, oder Ihre Arbeitsstation darf zugreifen. In beiden Fällen muß Ihnen jedenfalls der Zugang gewärt werden. Das bedeutet dann auch, daß Sie von anderen Arbeitsstationen auf Ihre Datenbank zugreifen können. Das birgt natürlich auch Gefahren, wenn Sie z.B. einen Host eintragen, und DHCP im Netzwerk aktiviert haben. Es ist dann nur eine Frage der Zeit, bis jemand rein zufällig Ihre Rechte nuten kann, weil das DHCP System diesem User Ihre alte IP-Nummer vergeben hat.

Die Privilegien einer Spalte wird also wie folgt bestimmt:

```
global privileges
ODER (database privileges AND host privileges)
ODER table privileges
ODER column privileges
```

Wenn Sie Rechte vergeben, dann bleiben diese solange aktiv, bis Sie diese mit dem **DELETE** Befehl aufheben. Das bedeutet im Klartext, daß Rechte,die mit **GRANT** vergeben wurden, nicht mit **REVOKE** gelöscht werden können. Hierzu muß explizit **DELETE** verwendet werden.

In MySQL 3.22.12 oder höher wird beim Anlegen eines Users ein bestimmtes Paßwort gesetzt, welches der **IDENTIFIED BY** Klausel entnommen wird. In diesem Fall wird das alte Paßwort durch ein neues ersetzt.

Aber Achtung: Wenn keine solche Klausel angeben wird, dann besitzt der User kein Paßwort. Das sollte nie passieren.

Paßworte werden mit der Option **SET PASSWORD** gesetzt. Siehe Kapitel SET.

Wenn GRANT Privilegien für eine Datenbank vergeben werden, wird ggf. eine mysql.db Tabelle angelegt. Erst, wenn alle Einträge mit REVOKE gelöscht wurden, wird diese Tabelle wieder gelöscht.

Wenn ein User keinerlei Privilegien an einer Tabelle besitzt, dann wird diese auch nicht angezeigt, z.B. wenn der User sich eine Liste der Tabelle mit **SHOW TABLES** anzeigen läßt. Diese Tabelle ist dennoch vorhanden.

Mit der Option **WITH GRANT OPTION** Klausel kann man einem User die Erlaubnis geben, seine Rechte auch an andere User weiterzugeben. Daher sollte man mit dieser Option sehr vorsichtig sein.

Man kann also keine Rechte weitervererben, die man selber nicht besitzt. Man sollte jedoch aufpassen, wenn man einem User Rechte in dem Moment vererbt, wenn man selber gerade höhere Privilegien besitzt. Um dies zu verhindern, muß man sich entweder ausloggen und unter einem neuen Level einloggen, oder man verwendet die Option **WITH GRANT**, um die Rechte für eine Datenbank einzuschränken, die man gerade einem anderen User vererben möchte. Wenn Sie also das **UPDATE** Recht vergeben, dann erhält der User automatisch auch das Recht für **INSERT** und **SELECT**.

Man sollte normalerweise keinem User das **ALTER** Recht geben, da ansonsten ein User die Rechte verändern kann, indem er eine Tabelle umbenennt.

Beim Start von MySQL werden alle Privilegien in das RAM geladen. Damit werden alle Regeln aktiv, sobald sich ein User an die Datenbank anbindet. Alle Modifikationen an den Rechten, die mit GRANT oder REVOKE ausgeführt werden, werden von dem User direkt bemerkt. Wenn Sie jedoch per Hand die Rechte verändern, z.B. mit INSERT oder UPDATE, dann müssen die unbedingt alle Privilegien mit **FLUSH PRIVILEGES** aktualisieren. Dies können Sie entweder über ein SQL Statement erreichen, oder Sie verwenden das Toolkit **mysqladmin flush-privileges**.

Die größten Unterschiede zwischen ANSI SQL und MySQL bei GRANT sind diese:

- ANSI SQL besitzt keine globalen oder auf Datenbanken bezogene Privilegien. MySQL kann da viel mehr.
- Wenn eine Tabelle in MySQL gelöscht wird, werden automatisch auch alle Rechte gelöscht (REVOKE).
- Alle vererbten Rechte werden ebenfalls gelöscht.






30. ODBC Treiber für MySQL

Die ODBC-Schnittstelle definiert folgendes:

- eine Bibliothek mit ODBC-Funktionsaufrufen, mit denen eine Verbindung zur Datenquelle hergestellt, SQL-Anweisungen ausgeführt und Ergebnisse abgerufen werden;
- eine Standardmethode zur Herstellung einer Verbindung zur und Anmeldung bei der Datenquelle;
- eine auf der X/Open und SQL Access Group (SAG) CAE-Spezifikation (1992) aufbauende SQL-Syntax;
- eine standardisierte Darstellungsweise für Datentypen;
- eine standardisierte Gruppe von Fehlercodes.







30.1 ODBC-Treibertypen

Man gruppiert ODBC Treiber nach einschichtigen und mehrschichtigen Treibern:

- Ein **einschichtiger ODBC - Treiber** (Single-Tier) verarbeitet sowohl ODBC-Funktionen als auch SQL-Anweisungen. In andern Worten: ein einschichtiger ODBC-Treiber umfaßt die Datenzugriffs - Software, die zur Verwaltung der Datenquellendatei und Katalogtabellen erforderlich ist. Einschichtige ODBC-Treiber werden beispielsweise zum Zugriff auf Xbase-Dateien verwendet. Ein Beispiel ist der dBASE-Treiber.
- Ein **Mehrschichtiger ODBC - Treiber** (Multiple-Tier) verarbeitet ODBC-Funktionen, sendet SQL-Anweisungen zur Verarbeitung aber an die Datenbank - Maschine. Im Gegensatz zu einem einschichtigen Treiber umfaßt ein mehrschichtiger Treiber keine Datenzugriffssoftware zur direkten Verwaltung der Daten. Der Watcom SQL-Treiber ist z.B. ein mehrschichtiger ODBC-Treiber.







30.2 ODBC Kompatibilitätsstufen

Wenn Sie mit QBase oder InfoMaker arbeiten, können Sie zum Datenzugriff ODBC-Treiber von anderen Herstellern einsetzen, wie z.B. direkt von DBMS-Anbietern bezogene Treiber. Wenn Sie einen ODBC-Treiber eines Drittherstellers verwenden möchten, muß dieser bestimmte Kompatibilitätsanforderungen erfüllen, so daß er beim Einsatz zusammen mit QBase oder InfoMaker ordnungsgemäß funktioniert.

QBase und InfoMaker können auf zahlreiche Datenquellen zugreifen, für die ODBC-kompatible Treiber erhältlich sind. Von Drittherstellern vertriebene ODBC-Treiber können sich vom Funktionsumfang her jedoch stark voneinander unterscheiden.

Damit ein standardisiertes Maß an Kompabilität mit der ODBC-Schnittstelle gewährleistet ist und Anwendungshersteller kontrollieren können, ob ein bestimmter Treiber die erforderlichen Funktionen aufweist, sind in der ODBC-Spezifikation Kompatibilitätsstufen für die folgenden zwei Bereiche definiert:

API umfaßt die unterstützten ODBC-Funktionsaufrufe; **SQL-Grammatik** umfaßt die unterstützten SQL-Anweisungen und SQL-Datentypen.







30.3 API-Kompatibilitätsstufen

ODBC definiert vom Funktionsumfang her drei API-Kompatibilitätsstufen:

- Kern (Core) umfaßt eine Gruppe von API-Kernfunktionen, die den Funktionen in der X/Open und SAG Call Level Interface (CLI) Spezifikation entsprechen.
- Stufe 1 (Level 1) umfaßt alle API-Kernfunktionen sowie mehrere erweiterte Funktionen.
- Stufe 2 (Level 2)umfaßt neben den Kernfunktionen und den Funktionen der Stufe 1 noch zusätzliche erweiterte Funktionen. Stellen Sie sicher, daß der mit QBase oder InfoMaker verwendete ODBC-Treiber den API-Kompatibilitätsanforderungen für Stufe 1 oder höher entspricht.







30.4 SQL-Kompatibilitätsstufen

ODBC definiert vom Funktionsumfang her drei SQL-Grammatik-Kompatibilitätsstufen:

- **Minimum**, eine Gruppe von SQL-Anweisungen und -Datentypen, die ein Mindestmaß an ODBC-Kompatibilität gewährleisten.
- **Kern (Core)** umfaßt alle SQL-Grammatikfunktionen der Stufe "Minimum" sowie zusätzliche Anweisungen und Datentypen, die weitgehend der X/Open und SAG CAE-Spezifikation (1992) entsprechen.
- **Erweitert (Extended)** umfaßt alle SQL-Grammatikfunktionen der Stufen "Minimum" und "Kern (Core)" sowie erweiterte Anweisungen und Datentypen, die gängige DBMS-Erweiterungen für SQL unterstützen.






30.5 Installation des MyODBC-Treibers für ACCESS

Dieses Kapitel beschreibt die Realisierung des ODBC-Zugriffs auf eine - Datenbank von einem Windows-Client aus. Hier erfolgt derZugriff auf die Datenbank über das TCP/IP Protokoll im LAN oder auch über das INTERNET.

Die verwendete Datenschnittstelle nennt sich ODBC (Open Database Connectivity). Dabei handelt es sich um einen Standard für Datenbankzugriffe der ursprünglich von Microsoft stammt. Im Normalfall kann nämlich eine Datenbank-Applikation (in diesem Beispiel Microsoft Access) nicht auf fremde Datenbank-Formate zugreifen (von einigen Ausnahmen abgesehen). Abbildung Access

Die ODBC-Schnittstelle ist nun eine Möglichkeit, dieses Problem zu lösen. Dazu wird die Datenbank auf die nicht direkt zugegriffen werden kann, hinter einem ODBC-Treiber "versteckt". Der eingesetzte Treiber ist für das jeweilige Datenbankformat maßgeschneidert und somit in der Lage auf die Datenbank zugreifen zu können. Seiner Umgebung, also dem Betriebssystem, bietet der ODBC-Treiber solche Datenbanken nun als ODBC-Datenquellen an. ODBC-Datenquellen erlauben einen standardisierten Zugriff mit Hilfe einer genau definierten Schnittstelle. Der jeweilige ODBC-Treiber setzt die über diese Schnittstelle eintreffenden Kommandos in das spezifische Format der verwendeten Datenbank um. Die eigentliche Datenbank-Applikation muß nun nur noch in der Lage sein auf ODBC-Datenquellen zugreifen zu können.

Fazit: Im System muß ein auf die jeweilige Datenbank angepaßter ODBC-Treiber installiert sein.

Abblildung Access2

Die andere Variante ist ein direkter Zugriff mit Hilfe von MySQL-Frontends. Diese sind jedoch hauptsächlich für den UNIX-Bereich verfügbar. Die Tools für den Windows-Bereich sind momentan noch eher dürftig. Sie können sich jedoch jederzeit im Download- bzw. Contribution-Bereich des Datenbankherstellers TcX umsehen ob zwischenzeitlich komfortablere Tools zur Verfügung stehen.

Installation des ODBC-Treibers unter Windows 95 (auch für Win98/NT) Da insbesondere der ODBC-Zugriff über eine Microsoft-Windows-Oberfläche relevant sein dürfte, anbei eine Erläuterung der einzelnen Schritte für Windows95 (für Windows98 und WindowsNT dürften die einzelnen Schritte prinzipiell identisch sein).

Die notwendige ODBC-Software können sie wieder von der Download-Seite des Datenbankherstellers TcX herunterladen. Beachten sie aber, daß es für Windows95 und WindowsNT zwei verschiedene Varianten gibt. Besorgen sie sich die für ihr Betriebssytem passende Form. Alternativ dazu können sie die Software auch direkt von diesem Server downloaden:

Version 2.50.24 für Windows 95 (1328kB)

Version 2.50.24 für Windows NT (1512kB)

Nach dem Auspacken des ZIP-Archivs (und Lesen der README-Datei) starten sie bitte das

Setup-Programm. Über den Startbildschirm

Abbildung myodbc1

gelangen sie zur Treiberauswahl.

Abbildung myodbc2

Sie erhalten nur die Möglichkeit den MySQL-ODBC-Treiber zu installieren. Mit der Schaltfläche OK können sie einfach loslegen und ausprobieren ob die Installation funktioniert. Insbesondere wenn sie aber schon einen ODBC-Treiber-Manager installiert haben (z.B. weil sie bei Installation von Microsoft Access alle Komponenten ausgewählt haben) kann es bei der Installation zu Konflikten kommen. So ließ sich auf meinem Rechner die Datei MFC30.DLL nicht überschreiben. In diesem Fall wählen sie vor Auswahl der OK-Schaltfläche die Advanced-Schaltfläche und gelange so in die "Advanced Installation Options". Hier wählen sie dann in diesem Fall "Do not install Driver Manager" und versuchen dann die Installation erneut. Zumindest auf meinem Rechner lief die Installation dannach problemlos durch.

Abbildung myodbc3

Nach der Installation öffnen sie die Systemsteuerung ihres Rechners

Abbildung system1

und öffnen mit einem Doppelklick auf das "32-Bit-ODBC"-Icon den ODBC-Datenquellen-Administrator.

Abbildung system2

Die Installationsroutine hat ihnen schon eine Datenquelle installiert ("sample MySQL") die allerdings auf keine reale Datenbank verweist. Sie können diese Datenquelle aus ihrem System entfernen. Über die Schaltfläche "Hinzufügen" gelangen sie zum Dialogfenster "Neue Datenquelle erstellen".

Hier wählen sie den MySQL-Treiber aus und betätigen anschließend die Schaltfläche "Fertigstellen". Sie gelangen dann in ein Dialogfenster zur Konfiguration der Datenquelle.

Hier sind die folgenden Parameter einstellbar:

Windows DNS name: Hier können sie entsprechend ihren eigenen Wünschen einen Namen für die Datenquelle vergeben. Unter der gewählten Bezeichnung erreichen sie später im System (z.B. unter Access) diese Datenquelle. Als Server (Datenbank-Server) geben sie den Namen oder die IP-Nummer ihres MySQL-Datenbank-Servers an.

MySQL Database name: Hier handelt es sich um den realen Namen der Datenbank auf dem MySQL-Server.

User: Hierbei handelt es sich um ihre Userkennung bei der Anmeldung am Datenbankserver. Beachten sie, daß es sich hierbei um die MySQL-Userkennung handelt.

Password: Das zur Userkennung paßende Passwort. Geben sie ihr Kennwort nie bekannt! Tragen sie es auch hier nie ein! Sie werden bei Zugriffsversuchen dann vom System nach dem Password gefragt! Port: Kann leer bleiben da der MySQL-Datenbankserver tatsächlich auf Port 3306 läuft.

Die verschiedenen Frontends (z.B. Access) implementieren verschiedene Versionen des ODBC-Standards oder weisen Implementierungsfehler auf. Je nach Frontend können sie deshalb noch

verschieden Optionen zum Verhalten des MySQL-ODBC-Treibers (MyODBC) einstellen. Lesen sie hierzu nochmals in der README-Datei des MyODBC-Pakets nach. Für Access 7.0 ist z.B. die Option "Return matching rows", für Access 2.0 die Optionen "Return matching rows" und "Simulate ODBC 1.0" zu setzen.

Verlassen sie anschließend den Dialog zur Konfiguration der MySQL-Datenbankquelle über die Schaltfläche OK.

Die soeben konfiguriert Datenquelle erscheint nun in der Liste der Benutzer-Datenquellen im ODBC-Datenquellen-Administrator. Die Datenbank auf ihrem MySQL-Datenbank-Server steht ihnen nun als ODBC-Datenquelle in ihrem System zur Verfügung.

Mit einem ODBC-fähigen Frontend können sie nun auf die Datenbank zugreifen und Datensätze lesen, schreiben, verändern sofern sie auf dem MySQL die entsprechenden Rechte haben. Die diversen Frontends können hier nicht umfaßend behandelt werden. Lesen sie notfalls in der zugehörigen Software-Dokumentation nach wie sie auf ODBC-Datenquellen zugreifen können. Nachfolgend wird nur für Microsoft-Access der Zugriff exemplarisch dargestellt.

Sie benötigen für die nachfolgenden Schritte eine funktionierende Netzwerk-Verbindung zu ihrem MySQL-Datenbank-Server! Erstellen sie sich unter Access eine neue, leere Datenbank. Anschließend muß eine Verknüpfung zu den Datenbank-Tabellen auf dem MySQL-Datenbank-Server erstellt werden. Wählen sie dazu in der Menüleiste "Datei"/"Externe Daten"/"Tabellen verknüpfen..." aus.

Im nun folgenden Auswahldialog wählen sie als Dateityp "ODBC-Datenbanken()" aus.

Nach dieser Auswahl erscheint sofort ein Dialog zur Auswahl der entsprechenden Datenquelle. Gehen sie hier zuerst zur zweiten Registerkarte "Computer-Datenquelle". In der dortigen Liste sollte die von ihnen konfigurierte ODBC-Datenquelle erscheinen. Wählen sie diese aus und beenden sie den Dialog mit Hilfe der Schaltfläche "OK".

Ihr System meldet sie nun am MySQL-Datenbankserver an und baut eine Verbindung zur konfigurierten Datenbank auf. Nachdem dies erfolgt ist, erhalten sie eine Liste der verfügbaren Tabelle(n). Wählen sie die Tabelle(n) aus auf die sie zugreifen möchten und verlassen sie auch diesen Dialog mit der Schaltfläche "OK".

Falls in der Tabelle kein Primärschlüssel definiert ist, müssen sie noch einen derartigen Primärschlüssel auswählen. Im nachfolgenden Beispiel wäre das Datenfeld "lfb_nr" auszuwählen.

Nach Verlassen dieses Dialogs haben sie im Access-Datenbank-Fenster die entsprechende Tabelle zur Verfügung. Sie ist mit dem Icon einer ODBC-Datenquelle gekennzeichnet.

Sie haben nun auf die zugehörigen Daten ganz normalen Zugriff wie auf lokale Tabellen auch. Insofern können Abfragen oder Formularfenster erstellt werden oder auch einfach mit Hilfe der Datenblattansicht zugegriffen werden.






30.6 MyODBC

MyODBC ist ein 32 Bit Level 0 ODBC-Treiber für Microsoft Windows Clients, die insbesondere speziell für MySQL angepasst wurden. Somit ist es möglich, mit Anwendungsprogrammen, wie Winword, Excel, Access, Staroffice, Applixware Office, u.s.w. auf die **MySQL** Datenbank zuzugreifen. Um die Sourcen der ODBC DLL mittels Microsoft VC++ zu compilieren, wird das ODBC 3.0 Package von Microsoft oder VC++ 5.0 Professional Edition benötigt. Diese Version wurde mittels VC++ 5.0 mit der Einstellung für ODBC 2.5 Definitionen kompiliert. Das in der Source-Distribution enthaltene Beispiel-Makefile und die Resources wurden für Microsoft VC++ 5.0 erstellt.







30.7 Kompatibilität zu Anwendungsprogrammen

MyODBC wurde erfolgreich mit Access, Admndemo.exe, C++-Builder, Centura Team Developer (früher Gupta SQL/Windows), ColdFusion (Solaris), Crystal Reports, DataJunction, Delphi, Excel, iHTML, FileMaker Pro, FoxPro, Notes 4.5/4.6, SBSS, Perl DBD-ODBC, Paradox, Powerbuilder, VC++ und Visual Basic getestet. Dennoch gibt es einige bekannte Probleme mit ODBC Treibern und Anwendungsprogrammen.




30.8 Bekannte Probleme mit MyODBC

Alle Programme arbeiten zuverlässig mit MyODBC, es gibt jedoch ein paar Probleme, die häufiger auftreten. Die Gründe für diese Probleme werden hier nun kurz diskutiert:

- **ACCESS**
  - Anbindungen können scheitern, wenn nicht zumindest ein Primary Key in jeder Tabelle ist.

    Damit Updates auch erfolgreich durchgeführt werden können, muß ein TIMESTAMP in der jeweils zu aktualisierenden Tabelle enthalten sein.
  - Es sollte nur Felder mit double float verwendet werden. Bei Vergleichen von single float Werten produziert ACCESS Fehler. Das Feld "return matching rows" im "option field" muß gesetzt sein, wenn Access sich an MySQL anbindet.
  - Access auf Windows NT 4.0 interpretiert die BLOB Spalten als OLE Objekte. Diese können durch MEMO Spalten ersetzt werden, indem man die Spalten in MySQL mit Hilfe von ALTER TABLE zu TEXT umdefiniert.
  - ACCESS kann nicht immer DATE Spalten korrekt behandeln. Hier sollten die Spalten auf das DATETIME Format gesetzt werden.
  - In einigen Fällen erzeugt ACCESS illegale SQL-Statements, die MySQL nicht verstehen kann. Damit die SQL-Befehle auch 1:1 durchgereicht werden, muß man unter ACCESS im Menü **Query|SQLSpecific|Pass-Through**.
  - DataJunction: Man sollte bei dem Export von Daten den Ausgabetyp VARCHAR unbedingt ENUM vorziehen, da ACCESS hier Fehler produziert. Unter EXCEL ist dies ohne Probleme. Hier noch einige Tips: Bei Problemen mit Datum sollte man diese als Strings behandeln:

    ```
    select CONCAT(rise_time), CONCAT(set_time)
        from sunrise_sunset;
    ```

    Sobald die Werte als Strings übergeben werden, werden diese von EXCEL97 korrekt interpretiert. Der Trick liegt darin, daß CONCAT den ODBC - Treiber täuscht und dieser die Uhrzeit als String übergibt. EXCEL konvertiert alle Strings in das Format TIME, ein BUG in EXCEL. ODBC Treiber über geben normalerweise den exakten Typ einer Spalte an das Anwendungsprogramm, im ODBC Treiber wurde aber so ein Workaround für den EXCEL BUG programmiert.
  - **odbcadmin** ist ein Testprogramm für ODBC.
  - **Borland / Inprise DELPHI** MySQL arbeitet nur mit DBE ab der Version 3.2 korrekt. Hier ein Beispiel Code für die Ansteuerung von MySQL mit DELPHI, welches ein ODBC Eintrag und ein BDE Eintrag für MyODBC generiert. Das Beispiel stammt von mailto:bryan@flesherfab.com:

    ```
    fReg:= TRegistry.Create;
      fReg.OpenKey('\Software\ODBC\ODBC.INI\DocumentsFab', True);
      fReg.WriteString('Database', 'Documents');
      fReg.WriteString('Description', ' ');
      fReg.WriteString('Driver', 'C:\WINNT\System32\myodbc.dll');
      fReg.WriteString('Flag', '1');
      fReg.WriteString('Password', '');
      fReg.WriteString('Port', ' ');
      fReg.WriteString('Server', 'xmark');
      fReg.WriteString('User', 'winuser');
      fReg.OpenKey('\Software\ODBC\ODBC.INI\ODBC Data Sources', True);
      fReg.WriteString('DocumentsFab', 'MySQL');
      fReg.CloseKey;
      fReg.Free;

      Memo1.Lines.Add('DATABASE NAME=');
    ```

```
Memo1.Lines.Add('USER NAME=');
Memo1.Lines.Add('ODBC DSN=DocumentsFab');
Memo1.Lines.Add('OPEN MODE=READ/WRITE');
Memo1.Lines.Add('BATCH COUNT=200');
Memo1.Lines.Add('LANGDRIVER=');
Memo1.Lines.Add('MAX ROWS=-1');
Memo1.Lines.Add('SCHEMA CACHE DIR=');
Memo1.Lines.Add('SCHEMA CACHE SIZE=8');
Memo1.Lines.Add('SCHEMA CACHE TIME=-1');
Memo1.Lines.Add('SQLPASSTHRU MODE=SHARED AUTOCOMMIT');
Memo1.Lines.Add('SQLQRYMODE=');
Memo1.Lines.Add('ENABLE SCHEMA CACHE=FALSE');
Memo1.Lines.Add('ENABLE BCD=FALSE');
Memo1.Lines.Add('ROWSET SIZE=20');
Memo1.Lines.Add('BLOBS TO CACHE=64');
Memo1.Lines.Add('BLOB SIZE=32');

AliasEditor.Add('DocumentsFab','MySQL',Memo1.Lines);
```

- **C++Builder** ist mit BDE 3.0 erfolgreich getestet. Das einzige Problem ist, daß wenn eine Tabellendefinition geändert wird, die Abfragefelder nicht automatisch mit aktualisiert werden. BDE kann offensichtlich keine **primary keys** erkennen. Eine Ausnahme ist **index primary**
- **Visual Basic** erfordert für eine Aktualisierung einer Tabelle, daß ein **primary key** definiert wurde.








30.9 Felder in dem ODBC Administrator

Es gibt mehrere Varianten, den Servernamen unter Microsoft Windows anzugeben. Es kann die IP-Nummer des Servers angegeben werden, oder auch ein DNS-Server aufgesetzt werden, der die Namensauflösung durchführt. Hier ein Beispiel des ODBC Setup´s:

```
Windows DSN name:   test
Description:        Meine Testdatenbank
MySql Database:     test
Server:             192.168.1.2
User:               testuser
Password:           mein_passwort
Port:
```

Der Port kann frei bleiben, sofern der alte Standardport (3306) von MySQL verwendet wird. Ab MySQL 3.23 wird ein neuer Standardport (3333) verwendet. Hierzu sehen Sie bitte im Firewall Handbuch unter SQL nach. Es ist wichtig, die Sicherheitsvorkehrungen alle genau zu beachten.

Der Wert für das DSN Feld ist irgendein eindeutiger Name, der für die ODBC Anbindungen unbedingt gebraucht wird. Ohne den Namen kann man verschiedene SQL-Datenbanken im Netzwerk nicht getrennt abfragen.

Falls die Optionen **Read Options** angegeben werden (C:\my.cnf), werden alle Daten aus dieser Datei geladen.







30.10 Wie man den Wert einer AUTO_INCREMENT Spalte in ODBC ausliest

Ein allgemeines Problem bei ODBC ist z.B. die ID eines INSERT Statements auszulesen. Mit ODBC kann das wie folgt geschehen (angenommen, auto sei ein AUTO_INCREMENT Feld):

```
INSERT INTO foo (auto,text) VALUES(NULL,'text');
SELECT LAST_INSERT_ID();
```

Oder, wenn die ID in eine andere Tabelle eingefügt werden soll:

```
INSERT INTO foo (auto,text) VALUES(NULL,'text');
INSERT INTO foo2 (id,text) VALUES(LAST_INSERT_ID(),'text');
```

Um von ODBC Anwendungsprogrammen profitieren zu können (DELPHI und ACCESS), hier ein Beispiel für eine Abfrage, die dem Auffinden einer neu eingefügten Reihe (Zeile) dient:

```
SELECT * FROM tbl_name WHERE auto IS NULL;
```







30.11 Optionen

Dem Treiber kann im Optionen-Feld mitgeteilt werden, daß der Client nicht 100% ODBC-Kompatibel ist. Jedes Bit dieses Flags hat eine bestimmte Bedeutung:

- **1** Der Client kann mit der realen Spaltenbreite, welche von MyODBC geliefert wird, nicht umgehen.
- **2** Der Client kann mit dem Rückgabewert einer Query, wieviele Datensätze bearbeitet wurden, nicht umgehen.
- **4** Erstellt ein Debug-Log in **C:\myodbc.log.** Hat die selbe Bedeutung wie MYSQL_DEBUG=d:t:O,c::\myodbc.log als Eintrag in der Autoexec.bat.
- **8** Keine Packetlänger für Ergebnissmengen und Parameter setzen.
- **16** Nicht nachfragen, selbst wenn dies vom Treiber verlangt wird
- **32** ODBC 1.0 Treiber simulieren.
- **64** Die Verwendung von Datenbanknamen in **database.table.column** ignorieren. Hierfür kann weiterhin die Alternative **table@database.column** (**MySQL**3.22) verwendet werden.
- **128** Die Verwendung eines ODBC-Manager-Cursors erzwingen (experimentel)
- **256** Die Verwendung eines Extended fetch entfernen (experimentel)
- **512** CHAR Felder auf volle Spaltenlänge abbilden.

Wenn mehrere Optionen gleichzeitig verwendet werden sollen, sind die obigen Flags entsprechend zu setzten. Beispiel:

Option 12 (4+8) setzt Debugging ohne Packetlängenlimitierung.







30.12 Mögliche Einstellungen für bekannte Clients

```
BDE-Anwendungen                 1 (gelegentlich auch 3)
Visual Objects                  1
Vision                          2
Active Server Pages             2
Access 7.0                      2
Access 2.0                      32 (32 +2)
ODBCETE32 + quiktest            513 (512 +1)
```

Die Default **MYODBC.DLL** wurde mit den Debugger Optionen kompiliert. In der Distribution befindet sich zusätzlich noch **MYODBC2.DLL** ,welche mit voller Optimierung kompiliert wurde. Bei Problemen sollte stets auf die DLL mit Debugging Informationen zurückgegriffen werden, welche mehr Logmeldungen und präzisere Informationen für den Bugreport ausgibt.

Sollten grundsätzliche Probleme mit der Verwendung von MyODBC auftreten, sollte zuerst versucht werden, das Programm **admndemo** ans Laufen zu bringen.







30.13 Bekannte ODBC Probleme und Ursachen

Wrong connection port:

Der Port ist 3333 für **MySQL** 3.20 und 3306 (default) für **MySQL** 3.21 oder größer.

Access denied for... :

Hier bitte im **MySQL**-Reference-Manual zum Thema **Privilege System** nachschauen.







30.14 Bekannte Bugs

- Access kann keine Felder des Typs Float in einer Tabelle ohne Timestamp verwenden. Mit Double-Feldern soll dies funktionieren.
- Da das Microsoft **ODBC** Setup nicht erkennen kann, um welches Betriebssystem es sich handelt, wird die Distribution in zwei Versionen bereit gestellt: eine für Win95 und eine für WinNT.
- Der Unterschied liegt in der Datei odbc.inf welche dem Setup mitteilt, ob es **CTL3D95.DL_** oder **CTL3DNT.DL_** als **CTL3D32.DLL** installieren soll.
- **ODBC 2.50** unterstützt die Verwendung von **BIGINT** (64Bit) Zahlen nicht ausreichend. Hierbei kann es zu Problemen kommen.
- **TIMESTAMP** nach **TIME** Konvertierung schlägt fehl.
- Binary-Felder werden als String und nicht als HEX-String zurückgeliefert.
- Einige Konvertierungen richten sich nicht nach dem **ODBC** Standard, was aber keine Auswirkungen auf die Anwendungen haben dürfte.
- Einige Anwendungen verlangen einen **Full Cursor Support** von **MyODBC**, welcher allerdings noch nicht komplett implementiert ist. In diesem Fall sollte die Anwendung zur Verwendung der **ODBC** Manager Cursor Library umgebogen werden.
- **ODBC**< unterstützt keine Null-Datumsangaben der Art **0000-00-00**. **MyODBC** konvertiert diese Datumsangaben nach NULL. Tabellenfelder, welche upgedatet werden sollen und Null-Datumsangaben erhalten können, sollten nicht mit NOT NULL deklariert werden.
- Da **MyODBC** zur Zeit keine eigene Cursor-Library unterstützt, kann es nur zusammen mit Anwendungen eingesetzt werden, welche die **ODBC**-Spezifikationen unterstützen. Einige Programme, meistens von Microsoft!! selbst, richten sich nicht nach den **ODBC**-Spezifikationen. Diese funktionieren nur mit Treiber-Cursors. Hier sollte versucht werden, die Anwendung durch Verwendung der Option **-Flag 128** auszutricksen.
- **MyODBC** wurde noch nicht vollständig ausgetestet. Besonders bei der Compilierung mit **-DTHREAD** kann es zu kleineren Problemen kommen.
- Für das **Change-Log** bitte die Datei **readme** zu Hilfe ziehen.







30.15 Installation des MyODBC Treibers unter Windows 98

Zuerst die Neuigkeiten:

MyODBC 2.50.18 besitzt nun eine Benutzer-Interface für verschiedene MyODBC Optionen. Die Einstellungen sind dann erforderlich, wenn der Client nicht 100% ODBC Kompatibel ist, wie z.B. die verschiedenen ACCESS Versionen, die zwar den Microsoft ODBC Standard unterstützen, jedoch immer unterschiedlich implemeniert sind. Anscheinend weiß bei Microsoft die Rechten Hand nicht, was die Linke tut. Wenn Sie sich einmal den ODBC Standard anschauen, sehen Sie, daß es hier unterschiedliche Klassen von ODBC Treibern gibt, ähnlich den JDBC Treibern. Das Angebot reicht hier von einfachem Durchreichen der SQL Statements bis hin zu einer vollständigken Konvertierung von Datentypen.

Der MyODBC Treiber bietet zur Wahrung der Kompatibilität eine Reihe von Optionen an, damit Ihr Client auf der Arbeitsstation keine Schwierigkeiten hat, auf die MySQL Datenbank zuzugreifen.

Hier zu werden Bits gesetzt, die alle verschiedene Bedeutungen besitzen:

```
  1     Der Client kann nicht mit der von MySQL zurückgegebenen, echten breite
        der Spalten umgehen.
  2     Der Client kann nicht den Wert der vom Select Statement
        zurückgegebenen Reihen verarbeiten. Stattdessen liefert MySQL die
        Zahl der gefundenen Reihen zurück. (> 3.21.14)
  4     C:\myodbc.log nimmt alle ODBC Fehler auf. Alternativ kann man auch
        MYSQL_DEBUG=d:t:O,c::\myodbc.log in die AUTOEXEC.BAT schreiben.
  8     Keine Paketbegrenzungen für Ergebnisse und Parameter
 16     Ein Prompt auf Fragen des Treibers unterdrücken
 32     Simulation eines ODBC 1.0 Treibers
 64     Ignoriere den Datenbanknamen in datenbank.tabelle.spalte (MySQL 3.22)
128     ODBC Cursors Manager aktivieren
256     Kein extended fetch von Reihen
512     CHAR Felder mit voller Länge übergeben
1024    SQLDescribeCol/( gibt die vollständigen Spaltennamen zurück
2048    Das Kompressionsprotokoll aktivieren (MySQL > 3.22)
4096    Space nach einem Funktionsnamen und vor einer "(" ignorieren
8192    Verbindung mit NAMED PIPES unter Windows NT
16384   LONGLONG Spalten in INT Spalten umwandeln, wenn die Applikation
        diese nicht handeln kann.
32768   Gebe "user" als table_qualifier und table_owner zurück
65536   my.cnf lesen
131072  Führe einige weitere Sicherheitsüberprüfungen durch
```

Z.B. muß man nun, wenn man den DEBUGGER aktivieren und die Betrenzung der Paketgröße ausschalten möchte, die Optionen 4+8 auswählen. Das bedeutet, daß man den Wert 12 (4+8) wählen muß.







31. MySQL Tutorial

Dieses Tutorial ist eine Einführung in SQL und natürlich speziell **MySQL**. Auch für User von **mSQL** sind viele Abschnitte nützlich, auch wenn **mSQL** nicht alle Befehle kennt. Der Client von **MySQL** ist ein interaktives Programm, um eine einfache Datenbank anzulegen und zu verwalten. Es stellt die Verbindung zum SQL-Server her, startet Abfragen und zeigt die Ergebnisse an. **MySQL** kann auch im **batch mode** gefahren werden. Er können Abfragen in eine Datei geschrieben werden, welche dann von der Reihe nach abgearbeitet werden. Beide Varianten werden hier beschrieben.

Um eine Liste von Optionen von **MySQL** zu erhalten, muß der Client mit der Option **--help** aufgerufen werden.

Es sollte klar sein, daß **MySQL** auf dem Server installiert ist, und daß der **MySQL** Server gestaret ist:

```
shell> mysql --help
```

Dieses Kapitel beschreibt detailliert den Prozeß des Aufsetzens und Benutzens der Datenbank. Später wird noch genauer beschrieben, wie man **MySQL** mit an ein WWW-Interface bindet. Wer eine bereits vorhandene Datenbank nutzen möchte, der kann den ersten Abschnitt überspringen.

Die Clients sind in der jeweiligen Distribution enthalten. Wen beispielsweise von Windows 95/98/NT aus eine Verbindung zu einem Server über das Intranet oder Internet aufbauen möchte, der findet die für Microsoft vorkompilierten Clients in der Binär-Distribution für WIN32.







31.1 Verbinden mit dem Server

Damit man sich mit dem Server verbinden kann, ist es notwendig, einen **MySQL** Username und ein Passwort zu haben. Falls der Server auf einer anderen Maschine läuft, als der, an der Sie gerade sind, muß der Hostname bekannt sein und natürlich eine Verbindung über die typischen Ports möglich sein (Firewall Einstellung).

Mit den richtigen Parametern sollte man nun in der Lage sein, eine Verbindung aufzubauen:

```
shell> mysql -h host -u user -p
Enter password: ********
```

Die ******** repräsentiert Ihr Paswort, welches bei Aufforderung einzugeben ist.

Wenn alles korrekt arbeitet, dann sollten Sie eine einführende Mitteilung nun auf dem Bildschirm sehen:

```
shell> mysql -h host -u user -p
Enter password: ********
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 459 to server version: 3.22.20a-log

Type 'help' for help.

mysql>
```

Einige **MySQL** Installationen erlauben es Usern, sich als "anonymous" User in einen Server einzuloggen. In diesem Falle genügt die folgende Eingabe ohne jedwelche Optionen:

```
shell> mysql
```

Nachdem Sie sich erfolgreich angebunden haben, können Sie mit QUIT oder Strg-D den Editor verlassen:

```
mysql> QUIT
Bye
```







31.2 Eingabe von Abfragen

Vergewissern Sie sich, daß Sie mit dem Server verbunden sind. Es wurde natürlich noch keine Datenbank ausgewählt, mit welcher Sie nun arbeiten könnten. Ein MySQL Server kann viele verschiedene Datenbanken für unterschiedlichste User mit unterschiedlichsten Zugriffsrechten beherrbergen. Hierzu kommen wir später

Wichtiger im Moment ist, daß Sie lernen, wie man Befehle absetzt, wie man Datenbanken anlegt, Daten in diese einlädt und Abfragen startet.

Dieses Kapitel beschreibt die einfachsten Grundlagen, wie man Eingaben tätigt

Hier nun eine einfache Eingabe, die den Server nach seiner Versionsnummer und dem Datum fragt:

```
mysql> SELECT VERSION(), CURRENT_DATE;
+--------------+--------------+
| version()    | CURRENT_DATE |
+--------------+--------------+
| 3.22.20a-log | 1999-03-19   |
+--------------+--------------+
1 row in set (0.01 sec)
mysql>
```

Die Ausgabe zeigt eine Reihe von Dingen:

- Eine Eingabe wird immer mit einem Semikolon abschschlossen
- Die Ausgabe
- Die Zahl der Reihen
- Die Zeit der Abfrage
- Befehlsoptionen werden mit Komma abgetrennt
- Die Ausgabe der Befehle erfolgt in Tabellen, entsprechend der Reihenfolge der Befehlsoptionen
- Nach der Ausgabe ist der Server wieder bereit

Folgende Befehle sind gleichwertig, es erfolgt keine Unterscheidung zwischen Groß- und Kleinschreibung:

```
mysql> SELECT VERSION(), CURRENT_DATE;
mysql> select version(), current_date;
mysql> SeLeCt vErSiOn(), current_DATE;
```

Eine weitere Abfrge, die zeigt, daß MySQL auch als Taschenrechner arbeitet:

```
mysql> SELECT SIN(PI()/4), (4+1)*5;
+-------------+---------+
| SIN(PI()/4) | (4+1)*5 |
+-------------+---------+
|    0.707107 |      25 |
+-------------+---------+
```

Diese Befehle bisher waren nun besonders kurze Einzeiler. Es können aber auch mehrere Befehle gleichzeitig in einer Zeile eingegeben werden:

```
mysql> SELECT VERSION(); SELECT NOW();
+--------------+
| version()    |
+--------------+
| 3.22.20a-log |
+--------------+

+---------------------+
| NOW()               |
+---------------------+
| 1999-03-19 00:15:33 |
+---------------------+
```

Wie man sieht, erfolgt die Ausgabe auf mehrere Befehle nacheinander.

Die Eingabe kann so lang sein, wie Sie wollen und durch mehrere "RETURN" in mehreren Zeilen erfolgen. Erst wenn MySQL ein Semikolon innerhalb der Eingaben erkennt, die mit "RETURN" nur bestätigt werden, werden die Befehle ausgeführt. Ein einfaches "RETURN" ohne Semikolon in der Eingabe bewirkt noch nichts. Hier eine weitere, mehrzeilige Eingabe:

```
mysql> SELECT
    -> USER()
    -> ,
    -> CURRENT_DATE;
+--------------------+--------------+
| USER()             | CURRENT_DATE |
+--------------------+--------------+
| joesmith@localhost | 1999-03-18   |
+--------------------+--------------+
```

Das Beipiel zeigt, daß durchaus mehrere Zeilen eingegeben werden können. Erst mit einem Semikolon wird der Befehl gültig und abgearbeitet. Diese Tatsache ist wichtig, wenn man z.B. an WWW-Interfaces arbeitet. Die PERL oder PHP Scripte machen im Prinzip nichts anderes als den Editor so zu bedienen, wie Sie es gerade auch tun.

Wenn Sie entscheiden, das ein Befehl nicht mehr ausgeführt werden soll, während Sie noch bei der Eingabe sind, so können Sie die Eingabe durch \c abbrechen:

```
mysql> SELECT
    -> USER()
    -> \c
mysql>
```

Es passiert recht häufig, daß man bei einer komplexeren Abfrage ein Anführungszeichen einfach übersieht. Man ist zuerst der Meinung, daß alles korrekt ist, gibt zum Abschluß das Semikolon ein und drückt "RETURN":

```
mysql> SELECT * FROM my_table WHERE name = "Smith AND age < 30;
    ">
```

Man sieht an dem Prompt, daß MySQL noch irgendetwas erwartet, obwohl ein Semikolon eingegeben wurde. In diesem Falle ist offensichtlich, daß das Semikolon anders interpretiert wird, als zuvor beschrieben.

Das Beste, was man dann tun kann, ist einfach \c einzugeben, damit mysql dann den Befehl abbricht.

An diesem Punkt gibt man dann folgendes ein:

```
mysql> SELECT * FROM my_table WHERE name = "Smith AND age < 30;
    "> "\c
mysql>
```

MySQL zeigt nun an, daß der Befehl abgebrochen wurde. Es erscheint der normale Promt.






31.3 Anlegen und Benutzen einer Datenbank

Wenn Sie sich mit dem Editor und dem Abbrechen von Befehlen bereits auskennen, dann ist es nun Zeit, um folgende Dinge zu lernen:

- Wie lege ich eine Datenbank an
- Wie lege ich eine Tabelle an
- Wie lade ich Daten in eine Tabelle
- Wie frage ich Daten aus einer Tabelle ab
- Wie benutze ich mehrfache Tabellen

Wenn MySQL installiert wurde, sind zumeist auch die Beispieldatenbanken in dem Datenbankverzeichnis von MySQL mit enthalten. Der SHOW Befehl zeigt uns an, welche Datenabanken verfügbar sind:

```
mysql> SHOW DATABASES;
+----------+
| Database |
+----------+
| mysql    |
| test     |
| tmp      |
+----------+
```

Wenn die test Datenbank vorhanden ist, können wir diese nun versuchen, anzusprechen:

```
mysql> USE test
Database changed
```

Der Befehl USE, ebenso wie QUIT erfordert kein Semikolon, es schadet aber auch nicht....Dafür muß der Befehl USE aber auch alleine stehen.

Wenn Sie diese Test Datenbank benutzen, sollten Sie sich darüber im Klaren sein, daß eventuell mehrere User mit einem anonymen Account Befehle für diese Datenbank absetzen können. Fragen Sie Ihren **MySQL** Administrator nach einem privaten Account zur Datenbank. Ihr Administrator (oder auch Sie selber, wenn Die die Datenbank selber installiert haben) muß dann einen Befehl absetzten, der allgemein dieses Format hat:

```
mysql> GRANT ALL ON menagerie.* TO your_mysql_name;
```

Der Befehl GRANT im Unterkapitel GRANT wirkt sich auf die Privilegien eines Users aus. Siehe

Kapitel Privilegien. Bei diesem Beispiel wird alles erlaubt, also Vorsicht !






31.4 Anlegen und Auswählen einer Datenbank

Wir legen eine Datenbank namens "menagerie" an. Diese kann viele Tabellen enthalten, welche alle von unterschiedlichen Usern genutzt werden können. Für die genauen Parameter sehen Sie bitte in der Sprachreferenz **MySQL** in Kapitel CREATE nach.

```
mysql> CREATE DATABASE menagerie;
```

Unter UNIX sind die Datenbanknamen "case sensitive", also sollte man unbedingt Groß-und Kleinschreibung beachten. Dies gilt nicht für SQL Schlüsselworte.

Das Anlegen einer Datenbank wählt diese nicht auch zwangsweise zur Abfrage aus. Es kann also passieren, daß Sie gerade eine Datenbank angelegt haben, jedoch noch in einer anderen Suchabfragen tätigen. Das Ganze hat einen Hintergrund: Es ist so möglich, auf einer Datenbankmaschine, also einem MySQL-Dämon hunderte bzw. tausende von Datenbanken mit vielen unterscheidlichen Usern anzulegen. Im Internet kann somit jeder User für seine Domain ein CGI-Interface (PERL, ePERL oder PHP) installieren und dort nach Belieben Datenbanken mit WWW-Interface der Allgemeinheit zur Verfügung stellen. Da die Zugriffsrechte mit Passwort geschützt sind, können auch keine Mitbenutzer der Datenbankmaschine irgendwie auf Ihre Tabelle zugreifen. Wenn viele CGI-BIN´s von verschiedensten Domains die Datenabnkmaschine benutzen, wird jeweils mit USE schnell die Datenbank gewechselt. Da MySQL horrend schnell ist (nicht zu vergleichen mit ORACLE oder MS-SQL), fällt der Multiuserbestrieb von den Antwortzeiten kaum auf. Man kann zudem durch weitere CPU´s und RAM die Datenbank fast beliebig skalieren.

Zurück zu unserem Tutorial. Wir wählen nun eine Datenbank aus:

```
mysql> USE menagerie
Database changed
```

Die Datenbank muß nur ein einziges Mal angelegt werden, jedoch jedesmal mit USE ausgewählt werden. Bevor man also seine Abfragen tätigen kann, muß immer nach einem Start des Client die Datenbank mit USE ausgewählt werden. Alle weiteren Abfragen beziehen sich dann immer auf diese Datenbank.

Wir loggen und nochmals ein:

```
shell> mysql -h host -u user -p menagerie
Enter password: ********
```

Man sollte hier unbedingt beachten, daß das Wort "menagerie" nicht das Passwort (-p) ist. "menagerie" ist die Datenbank ! Wer das Passwort in dem Befehl gleich mit angeben möchte, der muß dies so schreiben:

```
shell> mysql -h host -u user -pmeinpasswort menagerie
shell>
```

Dies ist nur ein Beispiel, Sie müssen also für host und user noch die gültigen Namen einsetzen. Für den Host können Sie auch die IP-Nummer eines Servers im Internet angeben.






31.5 Anlegen einer Tabelle

Das Anlegen einer Tabelle ist keine einfache Sache. An diesem Punkt ist aber wahrscheinlich Ihre Datenbank noch leer und unstrukturiert. Das wird folgender Befehl zeigen:

```
mysql> SHOW TABLES;
Empty set (0.00 sec)
```

Das Anlegen einer Datenbank muß sehr sorgfältig vorgenommen werden. Prinzipiell sind alle Änderungen nachträglich machbar, jedoch muß man schon etwas Erfahrung mit SQL und speziell auch **MySQL** haben, um dieses durchführen zu können. Daher sollte man sich alle Optionen des Befehls CREATE im Kapitel CREATE einmal genau anschauen.

Der schwierigere Teil der Aufgabe wird sein, zu bestimmen, welche Struktur die Datenbank haben soll, welche Tabellen benötigt werden, und welche Spalten darin enthalten sein sollen. Dinge, wie die Indexierung können zu einem späteren Zeitpunkt vorgenommen werden. An dieser Stelle sei noch erwähnt, daß Benutzer von ACCESS mit Hife eines Makro´s die Tabellenstruktur von ACCESS nach **MySQL** konvertieren lassen können. Siehe auch Kapitel Links zu MySQL. Danach kann einfach über Export der Datenbank aus ACCESS und einen Import in **MySQL** die Daten übertragen werden. Dies kann auch über eine Netzwerkverbindung, also auch über ISDN, erfolgen.

Legen wir nun eine Beispieldatenbank für Tiere an:

Wir möchten eine Tabelle mit je einem Eintrag für jedes Tier. Wir nennen diese einfach "tier". Als Minimun sollte der Name des Tieres enthalten sein. Da der Name selber noch keine interessante Datenbank darstellt, nehmen wir noch den Besitzer, die Spezies, Geschlecht u.s.w. hinzu.

Was ist mit Alter ? Könnte sicher interessant sein, ist aber problematisch, da die Tiere ja älter werden. Das würde bedeuten, daß an jedem Geburtstag des Tieres der Eintrag für Alter um 1 erhöht werden müsste. Besser ist es also, gleich das Geburtsdatum anzulegen, und das Alter **MySQL** selber errechnen zu lassen. **MySQL** besitzt Berechnungsfunktionen aller Art, darunter auch eine Funktion für die Berechnung der Tage zwischen zwei Datumsangaben.

Legen wir nun also die Datenbank an:

```
mysql> CREATE TABLE tier (name VARCHAR(20), besitzer VARCHAR(20),
    -> spezies VARCHAR(20), geschlecht CHAR(1), geboren DATE, gestorben DATE);
```

Warum haben wir den Typ VARCHAR ausgewählt ? Die Einträge können in ihrer von uns als ausreichend angesehenen Länge eventuell irgendwann einmal doch zu kurz sein. Die Spaltenwerte dürfen bei der Angabe VARCHAR schwanken, und zwar von 1 bis 255. Beim Anlegen der Tabelle trifft man also eine Annahme, welche man später dann mit dem ALTER TABLE Statement korrigiert, siehe Kapitel ALTER.

Das Geschlecht kann einfach mit m und w angegeben werden, es genügt also ein Buchstabe. Wem die Syntax von CREATE noch nicht so klar ist, der findet den Befehl im Kapitel CREATE ausführlich erklärt.

Der Gebrauch des Typs DATE sollte klar sein.

Wir haben nun eine Tabelle angelegt:

```
mysql> SHOW TABLES;
+---------------------+
| Tables in menagerie |
+---------------------+
| tier                |
+---------------------+
```

Um zu überprüfen, daß unsere Tabelle auch wie gewünscht angelegt wurde, können wird den Befehl DESCRIBE verwenden:

```
mysql> DESCRIBE tier;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| name       | varchar(20) | YES  |     | NULL    |       |
| besitzer   | varchar(20) | YES  |     | NULL    |       |
| spezies    | varchar(20) | YES  |     | NULL    |       |
| geschlecht | char(1)     | YES  |     | NULL    |       |
| geboren    | date        | YES  |     | NULL    |       |
| gestorben  | date        | YES  |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
```

Sie können DESCRIBE auch dann verwenden, wenn Sie z.B. die Namen oder Typen Ihrer Spalten in einer Tabelle vergessen haben.






31.6 Laden von Daten in die Tabelle

Nachdem nun die Tabelle angelegt wurde, sollte wir sie bevölkern. Die Befehle LOAD DATA und INSERT kommen zur Anwendung. Zuvor sollten Sie eventuell im Kapitel LOAD DATA sich über die Eigenheiten mit Sonderzeichen in den ASCII Dateien informieren.

Angenommen, daß die tier Einträge folgendermaßen aussehen sollen. Man beachte das Format des Datums hier im Beispiel:

```
name            besitzer        spezies         geschlecht      geboren
gestorben
Fluffy          Harold          cat             w               1993-02-04
claws           Gwen            cat             m               1994-03-17
Buffy           Harold          dog             w               1989-05-13
Diane           Gwen            dog             m               1998-08-31
1995-07-29
```

Da wir mit einer leeren Tabelle arbeiten, sind viele Dinge einfacher. Legen wir also eine Textdatenbank als tier.txt an. Dies kann mit einem einfachen Editor erfolgen, z.B. dem Notepad (Bäh!) oder mit joe unter UNIX (LINUX).

Man sollte beachten, daß für fehlende Einträge diese auf NULL gesetzt werden müssen. Dies findet sich in der ASCII - Datei als \N wieder:

```
Whistler        Gwen            bird              \N              1997-12-09      \N
```

Sie können noch obige Einträge in der ASCII Datei anfügen, bevor wir diese dann in die Datenbank einlesen:

```
mysql> LOAD DATA LOCAL INFILE "tier.txt" INTO TABLE tier;
```

Oops, was soll die OPTION LOCAL hier ? Die Option LOCAL gibt an, daß sich die ASCII Datei tier.txt auf Ihrer Arbeitsstation befinden muß, und zwar in dem Verzeichnis, von welchem aus der Client gestartet wurde. Sie könne aber auch absolute Pfadangaben machen. Die Datei tier.txt wird ggf. über das Netzwerk auf den Server übertragen. Also Vorsicht bei großen Dateien (Siehe --compress Option beim Client)

Wir möchten nun einige Einträge hinzufügen, ohne den Editor und das Kommando LOAD DATA INFILE zu verwenden. Hierzu können wir im Client den Befehl INSERT INTO verwenden. Siehe auch Kapitel INSERT:

```
mysql> INSERT INTO tier
    -> VALUES ('Puffball','Diane','hamster','w','1999-03-30',NULL);
```

Unser Hamster lebt also noch. Für Hamsterfreunde siehe auch http://www.little-idiot.de/hamster/. Für Freunde von Kühen siehe auch http://www.little-idiot.de/bse/

Man sollte stets beachten, daß die Tiere in Anführungszeichen stehen müssen, und Leerfelder mit einem \N gekennzeichnet werden müssen, jedenfalls bei dem Befehl LOAD DATA INFILE. Bei dem Befehl INSERT ist die Syntax anders. Hier muß für ein leeres Feld NULL eingegeben werden.

Man sieht, daß das Laden aus einer ASCII-Datei mit weniger Tipparbeit verbunden ist, als der direkte Einsatz des Befehls INSERT INTO ...







31.7 Abfragen der Informationen aus der Tabelle

Der SELECT Befehl wird dazu verwendet, um Informationen aus der Tabelle zu lesen. Die allgemeine Systax ist:

```
SELECT was
FROM tabelle
WHERE bedingungen
```

was gibt an, was aus der tabelle wir sehen möchten, also welche Spalten. tabelle gibt an, welche Tabelle wir meinen und bedingungen gibt an, welche Zeilen wir unter welchen Bedingungen angezeigt sehen möchten. Probieren Sie es doch gleich live einmal aus: http://www.rent-a-database.de/mysql/mysqllive.html. Hier können Sie nichts zerstören oder versehentlich löschen. Voraussetzung ist, daß Sie JAVA in Ihrem Browser aktiviert haben. Viel Spaß !







31.8 Abfrage aller Daten

Die einfachste Form des Befehl SELECT ist es, alles ausgeben zu lassen:

```
mysql> SELECT * FROM tier;
+----------+-----------+---------+-------------+--------------+----------------+
| name     | besitzer  | spezies | geschlecht  | geboren      | gestorben      |
+----------+-----------+---------+-------------+--------------+----------------+
| Fluffy   | Harold    | cat     | w           | 1993-02-04   | NULL           |
| Claws    | Gwen      | cat     | m           | 1994-03-17   | NULL           |
| Buffy    | Harold    | dog     | w           | 1989-05-13   | NULL           |
| Fang     | Benny     | dog     | m           | 1990-08-27   | NULL           |
| Bowser   | Diane     | cat     | m           | 1998-08-31   | 1995-07-29     |
| Chirpy   | Gwen      | bird    | w           | 1998-09-11   | NULL           |
| Whistler | Gwen      | bird    | NULL        | 1997-12-09   | NULL           |
| Slim     | Benny     | snake   | m           | 1996-04-29   | NULL           |
| Puffball | Diane     | hamster | w           | 1999-03-30   | NULL           |
+----------+-----------+---------+-------------+--------------+----------------+
```

Freunde von Vögeln haben sicher auch Ihre Schwierigkeiten bei der Bestimmmung des Geschlechtes - von Vögeln ! Daher der Eintrag NULL

Diese Form von SELECT ist sicher sinnvoll, wenn man die ganze Tabelle ausgeben möchte.

Wer später mit den SQL Befehlen für Datumsfunktionen arbeiten möchte, dem sei jetzt schon gesagt, daß wird hier einen Fall von Wiedegeburt haben - die Katze Bowser ! Dies wird später bei der Bestimmung des Alters zu Problemen führen.

Das Problem können wir z.B. dadurch lösen, daß wird die ASCII-Datei tier.txt nochmals editieren und erneut einlesen.

```
mysql> DELETE FROM tier;
mysql> LOAD DATA LOCAL INFILE "tier.txt" INTO TABLE tier;
```

Wir mußten zuvor alles aus der Tabelle tier löschen. Der Befehl wird als DELETE * FROM tier interpretiert.

Da das ganze etwas unelegant ist, möchten wird gerne den Eintrag in der Tabelle updaten. Wir verwenden hierzu den Befehl UPDATE, zu finden im Kapitel UPDATE:

```
mysql> UPDATE tier SET geboren = "1989-08-31" WHERE name = "Bowser";
```







31.9 Auswahl spezieller Einträge

Die Einträge sind in den Reihen enthalten, die Felder in Spalten, ganz wie in EXCEL Tabellen.

Wir möchten uns gerne nur die Zeile von Bowser ausgeben lassen:

```
mysql> SELECT * FROM tier WHERE name = "Bowser";
+--------+----------+---------+-------------+--------------+----------------+
| name   | besitzer | spezies | geschlecht  | geboren      | gestorben      |
+--------+----------+---------+-------------+--------------+----------------+
| Bowser | Diane    | dog     | m           |  1989-08-31  |   1995-07-29   |
+--------+----------+---------+-------------+--------------+----------------+
```

Die Änderung wurde also ausgeführt, fein !

Stringvergleiche sind normalerweise **nicht** CASE SENSITIV, das Ergebnis bei der der Eingabe von "bowser" oder "BOWSER" ist iedntisch.

Ein weiteres Beispiel mit Datumsvergleich:

```
mysql> SELECT * FROM tier WHERE birth >= "1998-1-1";
+----------+----------+---------+-------------+--------------+-----------+
| name     | besitzer | spezies | geschlecht  | geboren      | gestorben |
+----------+----------+---------+-------------+--------------+-----------+
| Chirpy   | Gwen     | bird    | w           | 1998-09-11   | NULL      |
| Puffball | Diane    | hamster | w           | 1999-03-30   | NULL      |
+----------+----------+---------+-------------+--------------+-----------+
```

Es können auch Bedingungen mit einander verbunden werden:

```
mysql> SELECT * FROM tier WHERE spezies = "dog" AND geschlecht = "w";
+----------+----------+---------+-------------+--------------+-----------+
| name     | besitzer | spezies | geschlecht  | geboren      | gestorben |
+----------+----------+---------+-------------+--------------+-----------+
| Buffy    | Harold   | dog     | w           | 1989-05-13   | NULL      |
+----------+----------+---------+-------------+--------------+-----------+
```

Die vorhergehende Abfrage benutzt einen logischen Operator: AND. Hier nun die Anwendung für den Operator OR:

```
mysql> SELECT * FROM tier WHERE spezies = "snake" OR spezies = "bird";
+----------+----------+---------+-------------+--------------+-----------+
| name     | besitzer | spezies | geschlecht  | geboren      | gestorben |
+----------+----------+---------+-------------+--------------+-----------+
| Chirpy   | Gwen     | bird    | w           | 1998-09-11   | NULL      |
| Whistler | Gwen     | bird    | NULL        | 1997-12-09   | NULL      |
| Slim     | Benny    | snake   | m           | 1996-04-29   | NULL      |
+----------+----------+---------+-------------+--------------+-----------+
```

AND und OR können auch gemeinsam eingesetzt werden. Es ist immer eine gute Idee, viele Klammern zu verwenden. Sie erhöhen die Lesbarkeit bei komplexen Konstrukten und ermöglichen somit die schnellere Fehleranalyse. Ohne Klammern kann es passieren, daß die Hierarchie der Operatoren nicht mehr eindeutig zu verstehen ist und die **MySQL** keine eindeutig richtigen Ergebnisse liefert. Hier also das erste kompliziertere Beispiel:

```
mysql> SELECT * FROM tier WHERE (spezies = "cat" AND geschlecht = "m")
    -> OR (spezies = "dog" AND geschlecht = "w");

+----------+-----------+---------+-------------+--------------+----------------+
| name     | besitzer  | spezies | geschlecht  | geboren      | gestorben      |
+----------+-----------+---------+-------------+--------------+----------------+
| Claws    | Gwen      | cat     | m           | 1994-03-17   | NULL           |
| Buffy    | Harold    | dog     | w           | 1989-05-13   | NULL           |
+----------+-----------+---------+-------------+--------------+----------------+
```

Wie gesehen, werden immer alle Spalten ausgegeben. Es könnte jedoch sein, daß wir uns nur für wenige Spalten interessieren. Ein weiterer Grund, nur einige Spalten auszuwählen ist, daß Tabellen miteinander verknüpft werden können. Wir benutzen also die Ausgabe einer selektierten Spalte, um nach diesen Elementen in anderen Tabellen zu suchen. Theoretisch könnte man alle Elemente in einer riesigen Tabelle speichern, ähnlich einem EXCEL Sheet. Was ist aber, wenn man eine 3 Dimensionale Tabelle hat. Die Gefahr, daß viele Elemente leerbleiben und unnötig Speicher belegen ist groß. Eine 3-Dimensionale Tabelle kann man aber in zwei einfache Tabellen splitten und die Elemente über jeweils eine oder mehrere Spalten verknüpfen. Zum besseren Verständnis öffnen Sie ein EXCEL Sheet, und stellen sich vor, die Zellen wären kleine Parzellen eines großen Fußballfeldes. Die Spieler 1-11 laufen auf dem Feld herum. Tragen Sie ein, zu welcher Minute sich welcher Spieler auf welcher Parzelle befunden hat. Sie werden merken - man benötigt eine §-Dimensionale Konstruktion. Diese hat zudem sehr viele Freifelder mit NULL, da sich ein Fußballspieler ja zumeist nur in einer Parzelle momentan aufhält. Um einem riesigen EXCEL Sheet zu entkommen, kann man zwei Tabellen anlegen. Die eine mit den durchlaufend numerierten Parzellen (Nummer), Spieler und Minute und einer weiteren mit Nummer, Spalte und Zeile, damit man der Nummer auch X-Y Parzellen wieder zuordnen kann. Am Ende des Spiels sind beide Tabellen gefüllt, ohne daß Platz verschenkt wurde. Zurück also zur Auswahl von einzelnen Spalten:








31.10 Auswählen von Spalten

Wir wählen bei der Ausgabe nur zwei Spalten aus:

```
mysql> SELECT name, geburtstag FROM tier;
+----------+------------+
| name     | geburtstag |
+----------+------------+
| Fluffy   | 1993-02-04 |
| Claws    | 1994-03-17 |
| Buffy    | 1989-05-13 |
| Fang     | 1990-08-27 |
| Bowser   | 1989-08-31 |
| Chirpy   | 1998-09-11 |
| Whistler | 1997-12-09 |
| Slim     | 1996-04-29 |
| Puffball | 1999-03-30 |
+----------+------------+
```

Auch folgende Ausgabe ist recht einfach:

```
mysql> SELECT besitzer FROM tier;
+-----------+
| besitzer  |
+-----------+
| Harold    |
| Gwen      |
| Harold    |
| Benny     |
| Diane     |
| Gwen      |
| Gwen      |
| Benny     |
| Diane     |
+-----------+
<verb>
<p>
Wir haben ein kleines Problem. Die Besitzer werden gleich mehrfach
ausgegeben, wenn sie mehrere Tiere besitzen. Um dies zu verhindern, gibt es
das kleine Wörtchen DISTINCT:
<p>
<verb>

mysql> SELECT DISTINCT besitzer FROM tier;
+-----------+
| besitzer  |
```

```
+-----------+
| Benny     |
| Diane     |
| Gwen      |
| Harold    |
+-----------+
```

Fein, nun sehen wir keine Besitzer mehr doppelt ! Schauen wir uns folgendes Beispiel einmal an:

```
mysql> SELECT name, spezies, geboren FROM tier
    -> WHERE spezies = "dog" OR spezies = "cat";
+--------+---------+------------+
| name   | spezies | geboren    |
+--------+---------+------------+
| Fluffy | cat     | 1993-02-04 |
| Claws  | cat     | 1994-03-17 |
| Buffy  | dog     | 1989-05-13 |
| Fang   | dog     | 1990-08-27 |
| Bowser | dog     | 1989-08-31 |
+--------+---------+------------+
```

Die Reihenfolge der Anweisungen ist recht schwierig zu merken. Es geht aber, wenn man sich die Befehle einmal auf Deutsch vorliest:

Selektiere nach name, spezies, geboren aus (der Tabelle) tier, wobei spezies gleich hund oder die spezies auch katze (sein kann).

Es ist zwar kein gutes deutsch, aber verständlich.





31.11 Sortieren von Reihen

Bisher haben wir in den vorhergehenden Beispielen die Ausgabe in keiner besonderen Reihenfolge ausgeben lassen. Die Datenbank hat die Daten geliefert, wie sie gespeichert waren.

Um ein Ergebnis zu sortieren, kann der Befehl ORDER BY verwendet werden:

Hier nun die Geburtstage aller Tiere, sortiert nach Datum:

```
mysql> SELECT name, geboren FROM tier ORDER BY geboren ;
+----------+------------+
| name     | geboren    |
+----------+------------+
| Buffy    | 1989-05-13 |
| Bowser   | 1989-08-31 |
| Fang     | 1990-08-27 |
| Fluffy   | 1993-02-04 |
| Claws    | 1994-03-17 |
| Slim     | 1996-04-29 |
| Whistler | 1997-12-09 |
| Chirpy   | 1998-09-11 |
| Puffball | 1999-03-30 |
+----------+------------+

<verb>
<p>
Um in absteigender Reichenfolge zu sortieren, müssen wir einen weiteren
Ausdruck verwenden, DESC. Ohne diese Angabe wird immer aufsteigend sortiert,
daher erübrigt sich auch der Ausdruck ASC (für ASCEND):
<p><verb>

mysql> SELECT name, geboren FROM tier ORDER BY geboren DESC;
+----------+------------+
| name     | geboren    |
+----------+------------+
| Puffball | 1999-03-30 |
| Chirpy   | 1998-09-11 |
| Whistler | 1997-12-09 |
| Slim     | 1996-04-29 |
| Claws    | 1994-03-17 |
| Fluffy   | 1993-02-04 |
| Fang     | 1990-08-27 |
| Bowser   | 1989-08-31 |
| Buffy    | 1989-05-13 |
+----------+------------+

<verb>
<p>
```

Man kann auch nach mehreren Spalten sortieren. Wenn mehrere Treffer in einer Spalte gefunden und eingeordnet sind, dann werden diese noch nachsortiert. Ein Beispiel:<p>
<verb>

```
mysql> SELECT name, species, birth FROM tier ORDER BY species, birth DESC;
+----------+---------+------------+
| name     | spezies | geboren    |
+----------+---------+------------+
| Chirpy   | bird    | 1998-09-11 |
| Whistler | bird    | 1997-12-09 |
| Claws    | cat     | 1994-03-17 |
| Fluffy   | cat     | 1993-02-04 |
| Fang     | dog     | 1990-08-27 |
| Bowser   | dog     | 1989-08-31 |
| Buffy    | dog     | 1989-05-13 |
| Puffball | hamster | 1999-03-30 |
| Slim     | snake   | 1996-04-29 |
+----------+---------+------------+
```





31.12 Berechnung von DATUM

MySQL besitzt einige Funktionen, mit denen man mit Daten rechnen kann, z.B. um das Alter zu errechnen, oder um spezielle Datumsangeben auszusortieren. Berechnen wir also einfach mal das Alter unserer Tieren manuell nach einer einfachen Formel: Differenz in Tagen, dividiert durch 365. Es dürfte klar sein, daß hiermit wohl kaum Schaltjahre und die besonderen Schaltjahre alle 400 Jahre einbezogen sind. Jahr 2000 fest ist diese Formel als nicht. **MySQL** mit allen seinen Funktionen ist aber Jahr2000 fest, also keine Panik. Alle Datenbanken in der Industrie, die mit **MySQL** aufgebaut wurden, können also Silverster/Neujahr 2000 weiterlaufen. Viele Unternehmen in der Chemiebranche setzen diese z.B. für die Qualtiätsüberwachung ein. Hier für ist **MySQL** geradezu prädestiniert.

Ein Beispiel:

```
mysql> SELECT name, (TO_DAYS(NOW())-TO_DAYS(geboren))/365 FROM tier;
+----------+---------------------------------------+
| name     | (TO_DAYS(NOW())-TO_DAYS(geboren))/365 |
+----------+---------------------------------------+
| Fluffy   |                                  6.15 |
| Claws    |                                  5.04 |
| Buffy    |                                  9.88 |
| Fang     |                                  8.59 |
| Bowser   |                                  9.58 |
| Chirpy   |                                  0.55 |
| Whistler |                                  1.30 |
| Slim     |                                  2.92 |
| Puffball |                                  0.00 |
+----------+---------------------------------------+
```

Die Ausgabe hat nun einen kleinen Schönheitsfehler: Die nette Formel, die wir angegeben haben, erscheint dummerweise in der Ausgabe. Wir korrigieren das durch die Angabe von AS. AS wird auch Spalten Alias genannt:

```
mysql> SELECT name, (TO_DAYS(NOW())-TO_DAYS(geboren))/365 AS alter
    -> FROM tier ORDER BY name;
+----------+--------+
| name     | alter  |
+----------+--------+
| Bowser   | 9.58   |
| Buffy    | 9.88   |
| Chirpy   | 0.55   |
| Claws    | 5.04   |
| Fang     | 8.59   |
| Fluffy   | 6.15   |
| Puffball | 0.00   |
| Slim     | 2.92   |
| Whistler | 1.30   |
+----------+--------+
```

Wir haben nun ein schönes Ausgabeformat. Wer CGI-BIN´s schreibt oder mit PHP3 programmiert, der wird den Operator AS häufiger benutzen müssen. Die Zahl der Zeilen, die wiederverwendet werden können, steigt. Um die Ausgabe noch perfekt zu machen, müssen wir noch sortieren:

```
mysql>  SELECT name, (TO_DAYS(NOW())-TO_DAYS(geboren))/365 AS alter
    ->  FROM tier ORDER BY alter;
```

```
+----------+------+
| name     | alter|
+----------+------+
| Puffball | 0.00 |
| Chirpy   | 0.55 |
| Whistler | 1.30 |
| Slim     | 2.92 |
| Claws    | 5.04 |
| Fluffy   | 6.15 |
| Fang     | 8.59 |
| Bowser   | 9.58 |
| Buffy    | 9.88 |
+----------+------+
```

Man muß schon genau hinsehen, um festzustellen, daß man auch nach dem Alias sortieren kann.

Wie schaut es denn nun mit toten Tieren aus ? Wir haben eines in der Tabelle, Bowser. Lassen wir uns also alle Tiere ausgeben, die schon gestorben sind:

```
mysql>  SELECT name, geboren, gestorben, (TO_DAYS(gestorben)-TO_DAYS(geboren))/365 AS
alter
    ->  FROM tier WHERE gestorben IS NOT NULL ORDER BY alter;
+--------+------------+------------+------+
| name   | geboren    | gestorben  | alter|
+--------+------------+------------+------+
| Bowser | 1989-08-31 | 1995-07-29 | 5.91 |
+--------+------------+------------+------+
```

Die Abfrage benutzt den Ausdruck gestorben IS The query uses gestorben IS NOT NULL und nicht gestorben != NULL, weil NULL ein spezieller Ausdruck ist, nämlich ein Platzhalter. Während in der Mathematik sagen kann "nicht null" ist dies bei einem Platzhalter nicht so einfach möglich.

In der Paraxis könnten noch weitere Abfragen interessant sein, z.B. welche Tiere im nächten Monat Geburtstag haben. Für diese Abfrage sind Tag und Jahr ohne Bedeutung, da ja nur nach dem Monat gefragt ist. Die Konvertierung TO_DAYS(DATE) hatten wir stillschweigend eingeführt, nun führen wir einfach weitere Funktionen ein: YEAR(DATE), MONTH(DATE), DAY(DATE):

```
mysql> SELECT name, geboren, MONTH(geboren) FROM tier;
+----------+--------------+----------------+
| name     | geboren      | MONTH(geboren) |
+----------+--------------+----------------+
| Fluffy   | 1993-02-04   |              2 |
| Claws    | 1994-03-17   |              3 |
| Buffy    | 1989-05-13   |              5 |
| Fang     | 1990-08-27   |              8 |
| Bowser   | 1989-08-31   |              8 |
| Chirpy   | 1998-09-11   |              9 |
| Whistler | 1997-12-09   |             12 |
| Slim     | 1996-04-29   |              4 |
| Puffball | 1999-03-30   |              3 |
+----------+--------------+----------------+

<verb>
<p>
Auf diese Art kann man auch Geburtstage in einem bestimmten Monat finden:
<verb>

mysql> SELECT name, geboren FROM tier WHERE MONTH(geboren) = 5;
+-------+------------+
| name  | geboren    |
```

```
+-------+------------+
| Buffy | 1989-05-13 |
+-------+------------+

<verb>
<p>
Man kann auch nach Zeitinterwallen suchen:
<p>
<verb>

mysql> SELECT name, geboren FROM tier
    -> WHERE MONTH(geboren) = MONTH(DATE_ADD(NOW(), INTERVAL 1 MONTH));


mysql> SELECT name, geboren FROM tier
    -> WHERE MONTH(birth) = MOD(MONTH(NOW()),12) + 1;
```

Die Statements sind nicht alle identisch. Wenn man sich diese Ausdrücke einmal genauer ansieht, dann wird zuerst nach dem Zeitinterwall der nächten 29/30/31 Tage gesucht, im zweiten Beispiel wird mit dem Operator MODULO (MOD) gearbeitet. Es wird nach dem nächsten Monat gesucht, ab dem Ende diesen Monats.








31.13 NULL Werte

Der NULL Wert kann schon etwas verwirrend sein, wenn man nicht mit diesem vertraut ist. Vom Konzept her ist NULL ein Platzhalter. Der Ausdruck ! NULL würde also bedeuten "kein Platzhalter" (! bedeutet nicht). Ebenso bedeutet ja: "Person nicht da !" auch nicht, daß die Person nicht da ist, sondern daß sie woanders ist. Für neugierige einmal zum ausprobieren:

```
mysql> SELECT 1 = NULL, 1 != NULL, 1 < NULL, 1 > NULL;
+----------+-----------+----------+----------+
| 1 = NULL | 1 != NULL | 1 < NULL | 1 > NULL |
+----------+-----------+----------+----------+
|     NULL |      NULL |     NULL |     NULL |
+----------+-----------+----------+----------+
```

Es kommt nur Blödsinn heraus. Arbeitet man jedoch mit dem Operator IS NOT NULL dann ergibt sich folgendes:

```
mysql> SELECT 1 IS NULL, 1 IS NOT NULL;
+-----------+---------------+
| 1 IS NULL | 1 IS NOT NULL |
+-----------+---------------+
|         0 |             1 |
+-----------+---------------+
```

Dies bedeutet quasi, daß mit dem Ausdruck IS NOT NULL danach gefragt wird, ob ein Wert enthalten ist. Die Felder in SQL Datenbanken können drei Werte enthalten: 0, einen String oder eine Zahl, ein Datum oder auch **NULL**. Man spricht hier von **Dreiwertigkeit**.






31.14 Abfrage von Suchmustern (Patterns)

MySQL kennt alle Standards von ANSI SQL 92. Ein Beispiel, in welchem alle Namen gefunden werden sollen, die mit B anfangen:

```
mysql> SELECT * FROM tier WHERE name LIKE "b%";
+--------+----------+---------+------------+--------------+------------+
| name   | besitzer | spezies | geschlecht | geboren      | gestorben  |
+--------+----------+---------+------------+--------------+------------+
| Buffy  | Harold   | dog     | w          | 1989-05-13   | NULL       |
| Bowser | Diane    | dog     | m          | 1989-08-31   | 1995-07-29 |
+--------+----------+---------+------------+--------------+------------+
```

oder alle, die mit fy enden.

```
mysql> SELECT * FROM tier WHERE name LIKE "%fy";
+--------+----------+---------+------------+--------------+------------+
| name   | besitzer | spezies | geschlecht | geboren      | gestorben  |
+--------+----------+---------+------------+--------------+------------+
| Fluffy | Harold   | cat     | w          | 1993-02-04   | NULL       |
| Buffy  | Harold   | dog     | w          | 1989-05-13   | NULL       |
+--------+----------+---------+------------+--------------+------------+
```

Für alle, die ein "w" enthalten:

```
mysql> SELECT * FROM tier WHERE name LIKE "%w%";
+----------+----------+---------+------------+--------------+------------+
| name     | besitzer | spezies | geschlecht | geboren      | gestorben  |
+----------+----------+---------+------------+--------------+------------+
| Claws    | Gwen     | cat     | m          | 1994-03-17   | NULL       |
| Bowser   | Diane    | dog     | m          | 1989-08-31   | 1995-07-29 |
| Whistler | Gwen     | bird    | NULL       | 1997-12-09   | NULL       |
+----------+----------+---------+------------+--------------+------------+
```

Eine Besonderheit ist die Möglichkeit zur Angabe von genauen Wortlängen:

```
mysql> SELECT * FROM tier WHERE name LIKE "_____";
+----------+----------+---------+------------+--------------+------------+
| name     | besitzer | spezies | geschlecht | geboren      | gestorben  |
+----------+----------+---------+------------+--------------+------------+
| Claws    | Gwen     | cat     | m          | 1994-03-17   | NULL       |
| Buffy    | Harold   | dog     | w          | 1989-05-13   | NULL       |
+----------+----------+---------+------------+--------------+------------+
```

Über Patterns kann man viele Beispiele zeigen. Im folgenden werden nur Beispiele gebracht, von denen man

einige einmal ausprobieren sollte.

Um nach Pattern suchen zu können muß man das Kommando REGEXP oder NOT REGEXP anstelle von LIKE einsetzen. Hier nun ein paar Beispiele:

```
Um Namen zu finden, die mit b oder B beginnen :

mysql> SELECT * FROM tier WHERE name REGEXP "^[bB]";

Um Namen zu finden, die mit fy enden:

mysql> SELECT * FROM tier WHERE name REGEXP "fy$";

Um Namen zu finden, die W oder W enthalten:

mysql> SELECT * FROM tier WHERE name REGEXP "[wW]";

Um Namen zu suchen, die exakt 5 Buchstaben Länge haben:

mysql> SELECT * FROM tier WHERE name REGEXP "^.....$";

Derselbe Ausdruck mit Hilfe des "repeat" Operators:

mysql> SELECT * FROM tier WHERE name REGEXP "^.{5}$";
```






31.15 Zählen in Spalten

In der Praxis stellen sich oft immer wieder dieselben fragen. Grund können statistische Erhebungen sein. Auf unsere tier Datenbank bezogen, könnte man die Fragen stellen:

Wie oft kommt ein bestimmter Typ von Einträgen in einer Tabelle vor ? Wir fragen z.B. nach der Zahl der Tiere, der Zahl von Tieren je Besitzer, oder der Zahl der noch lebenden Tiere.....

Um diese Fragen alle beantworten zu können, müssen wir einige wichtige Informationen zu vor wissen. Z.B. wieviele Spalten es überhaupt in unserer Tabelle gibt. Hierfür benutzen wir den Operator COUNT():

```
mysql> SELECT COUNT(*) FROM tier;
+----------+
| COUNT(*) |
+----------+
|        9 |
+----------+
```

Ok, soweit so gut, der Operator COUNT(*) zählt also alle Spalten. Wir möchten aber z.B. wissen, wieviele Tiere ein Besitzer hat:

```
mysql> SELECT besitzer, COUNT(*) FROM tier GROUP BY besitzer;
+-----------+----------+
| besitzer  | COUNT(*) |
+-----------+----------+
| Benny     |        2 |
| Diane     |        2 |
| Gwen      |        3 |
| Harold    |        2 |
+-----------+----------+
```

Was zählen wir hier eigendlich ? Übersetzen wir das Statement einmal auf Deutsch:

Selektiere nach allen Besitzern, zähle alles aus der datenbank tier und gruppiere nach besitzer.

Nun auch keine große Hilfe !(mir ging es ebenso)

Erinnern wir uns an die frühern SELECT Befehle:

```
mysql> SELECT name, geboren FROM tier ORDER BY geboren DESC;
```

Stellen wir nun die neuen Befehle gegenüber:

```
mysql> SELECT besitzer, COUNT(*) FROM tier GROUP BY besitzer;
```

Der Hauptunterschied liegt darin, daß wir eine neue Ausgabespalte einführen (COUNT(*)), in der dann doppelte Zeilen gezählt und ausgegeben werden.

Man sollte hier niemals das GROUP BY Statement vergessen, ansonsten gibt es eine Fehlermeldung:

```
mysql> SELECT besitzer, COUNT(besitzer) FROM tier;
ERROR 1140 at line 1: Mixing of GROUP columns (MIN(),MAX(),COUNT()...)
with no GROUP columns is illegal if there is no GROUP BY clause
```

Um diese Fehlermeldung verstehen zu können, bedarf es etwas Einfühlungsvermögen in MySQL und den SQL Parser. Die Erfahrung bekommt man durch viel Übung. Hilfreich sind bei SQL-Datenbanken stets ausgiebige Fehlermeldungen, welche auch Aussagefähig sind. Obige Fehlermeldung stiftet mehr Verwirrung als sie hilfreich ist, dadurch, daß sie unbekannte Ausdrücke

ausgibt. Um durchzublicken, muß man wissen, wie ein SQL Parser vorgeht: Zuerst wird der Befehl SELECT interpretiert. Hier erfährt der Parser, daß die Spalten besitzer und noch eine weitere Spalte auszugeben ist, wobei alle Elemente dieser Spalte gezählt werden sollen (COUNT(*)). In der Klammer steht * für alle. Der Befehl COUNT() könnte auch noch andere Auswahlkriterien enthalten, außer (*). Woher soll nun der Parser wissen, welches die zweite Spalte ist, die zu sortieren ist ? FROM tier ist obligatorisch und gibt die Datenbank an. Es fehlt also ein Operator, der dem COUNT Befehl sagt, welche Spalte er zählen soll. Hierfür wurde dann der Operator GROUP BY eingeführt. COUNT() und GROUP BY gehören also immer zusammen. Der obige Fehler war, daß COUNT() in der Klammer ein Auswahlkriterium oder eine Bedingung erwartet, jedenfalls keinen Namen der Spalte. **MySQL** beanstandet, daß kein GROUP BY Ausdruck vorhanden ist. GROUP columns sind die Befehle (COUNT() oder auch MIN() oder MAX()). Zu NON GROUP columns zählt besitzer.

MySQL meint also, daß erstens der Ausdruck GROUP BY fehlt und hier nicht besitzer mit COUNT zusammengehört.

MySQL ist für die Qualität der Fehlermeldungen nicht gerade berühmt, andere SQL Datenbanken sind aber auch nicht viel besser.

COUNT() und GROUP BY sind nützlich, um Daten zu charakterisieren, also zu gruppieren und zu zählen. Die folgenden Beispiele zeigen, wie man aus einer Datenbank aussagefähige Statistiken gewinnt. Wir zählen einmal die Tiere je Spezies:

```
mysql> SELECT spezies, COUNT(*) FROM tier GROUP BY spezies;
+---------+----------+
| spezies | COUNT(*) |
+---------+----------+
| bird    |        2 |
| cat     |        2 |
| dog     |        3 |
| hamster |        1 |
| snake   |        1 |
+---------+----------+
```

Die Zahl der Tiere, nach Geschlecht geordnet:

```
mysql> SELECT geschlecht, COUNT(*) FROM tier GROUP BY geschlecht;
+-------------+----------+
| geschlecht  | COUNT(*) |
+-------------+----------+
| NULL        |        1 |
| w           |        4 |
| m           |        4 |
+-------------+----------+
```

In einem Falle war das Geschlecht nicht zu bestimmen....Nun zählen wir Tiere und sortieren nach Spezies und Geschlecht:

```
mysql> SELECT spezies, geschlecht, COUNT(*) FROM tier GROUP BY spezies, geschlecht;
+---------+-------------+----------+
| spezies | geschlecht  | COUNT(*) |
+---------+-------------+----------+
| bird    | NULL        |        1 |
| bird    | w           |        1 |
| cat     | w           |        1 |
| cat     | m           |        1 |
| dog     | w           |        1 |
| dog     | m           |        2 |
| hamster | w           |        1 |
| snake   | m           |        1 |
+---------+-------------+----------+
```

Was ist hier passiert ? Ganz einfach: Wir geben die Spalten Spezies, geschlecht aus und in der dritten Spalte zählen wir die doppelten Einträge der beiden Spalten.

Wenn wir nur Katzen und Hunde zählen wollen, brauchen wir natürlich die anderen Tiere nicht mit ausgeben lassen:

```
mysql> SELECT spezies, geschlecht, COUNT(*) FROM tier
    -> WHERE spezies = "dog" OR spezies = "cat"
    -> GROUP BY spezies, geschlecht;
```

```
+---------+-------------+----------+
| spezies | geschlecht  | COUNT(*) |
+---------+-------------+----------+
| cat     | w           |        1 |
| cat     | m           |        1 |
| dog     | w           |        1 |
| dog     | m           |        2 |
+---------+-------------+----------+
```

Ein langes Statement. Es hält sich aber an das, was wir bereits kennen. COUNT() und GROUP BY stehen da, wenn auch durch den WHERE Operator voneinander getrennt. Man sollte sich stets die Reihenfolge SELECT ... FROM ... WHERE genau merken. COUNT muß nach dem SELECT stehen und GROUP BY steht immer am Ende.

```
mysql> SELECT spezies, geschlecht, COUNT(*) FROM tier
    -> WHERE geschlecht IS NOT NULL
    -> GROUP BY spezies, geschlecht;
+---------+-------------+----------+
| spezies | geschlecht  | COUNT(*) |
+---------+-------------+----------+
| bird    | w           |        1 |
| cat     | w           |        1 |
| cat     | m           |        1 |
| dog     | w           |        1 |
| dog     | m           |        2 |
| hamster | w           |        1 |
| snake   | m           |        1 |
+---------+-------------+----------+
```

So, mit diesen Abfragemöglichkeiten kann man schon eine ganze Menge machen. Befriedigend ist unsere Tabelle jedoch nicht. Wir möchten weitere Informationen speichern und eventuell diese miteinander verknüpfen.








31.16 Verknüpfen von Tabellen

Welche weiteren Informationen brauchen wir ? Z.B Ereignisse, die mit Datum, Typ und Bemerkungen festgehalten werden. Darin könnte man z.B. Geburten von Nachkommen, Tierarztbesuche o.ä. festhalten. Nicht, daß wir nun eine komplette Datenbank für Tierärzte aufbauen möchten....man könnte sie aber dafür nehmen, wenn sie fertig ist.....Wir legen also ohne weitere Überlegungen eine neue Tabelle in der Datenbank an:

```
mysql> CREATE TABLE ereignis (name VARCHAR(20), datum DATE,
    -> typ VARCHAR(15), bemerkung VARCHAR(255));
```

Laden wir nun eine ASCII Tabelle ein:

```
Fluffy  1995-05-15      spitz   4 kittens, 3 weiblich, 1 männlich
Buffy   1993-06-23      spitz   5 welpen, 2 weiblich, 3 männlich
Buffy   1994-06-19      spitz   3 welpen, 3 weiblich,
Chirpy  1999-03-21      küken   gelb weiss
Slim    1997-08-03      Küken   gebrochene rippe
Bowser  1991-10-12      perser
Fang    1991-10-12      perser
Fang    1998-08-28              birthday
Claws   1998-03-17              birthday
Whistler1998-12-09              birthday
```

Laden wir nun die ASCII Tabelle in die Datenbank ein:

```
mysql> LOAD DATA LOCAL INFILE "ereignis.txt" INTO TABLE ereignis;
```

Wir haben nun zwei Datenbanken. Beide möchten wir zusammen benutzen. Hier also einfach einmal ein Beispiel für eine Abfrage über zwei Tabellen hinweg:

```
mysql> SELECT tier.name, (TO_DAYS(date) - TO_DAYS(birth))/365 AS alter, bemerkung
    -> FROM tier, ereignis
    -> WHERE tier.name = ereignis.name AND type = "spitz";
+--------+------+-----------------------------------+
| name   | alter| bemerkung                         |
+--------+------+-----------------------------------+
| Fluffy | 2.27 | 4 kittens, 3 weiblich, 1 männlich |
| Buffy  | 4.12 | 5 welpen, 2 weiblich, 3 männlich  |
| Buffy  | 5.10 | 3 welpen, 3 weiblich              |
+--------+------+-----------------------------------+
```

Einige Dinge bedürfen noch der Erklärung:

- Der FROM Operator besitzt nun zwei Tabellen
- Wenn man die Informationen aus mehreren Tabellen kombiniert, dann muß eine neue Syntax für Spaltennamen verwendet werden: tabelle.spalte
- Die Tiernamen stehen in der einen Tabelle unter name und in der anderen unter ereignis. Hier muß man genau unterscheiden.

Um Tabellen miteinander zu verknüpfen, benutzt man gewöhnlich den JOIN Befehl. Man muß aber nicht zwangsläufig zwei verschiedene Tabellen nutzen, um einen JOIN auszuführen. Manchmal kann es auch sinnvoll sein, Einträge in einer Tabelle mit anderen Einträgen in derselben Tabelle miteinander zu vergleichen. Das Schlüsselwort JOIN muß in diesem Beispiel auch nicht explizit angegeben werden:

```
mysql> SELECT p1.name, p1.geschlecht, p2.name, p2.geschlecht, p1.spezies
    -> FROM tier AS p1, tier AS p2
    -> WHERE p1.spezies = p2.spezies AND p1.geschlecht = "f" AND p2.geschlecht = "m";
+--------+------------+--------+------------+---------+
| name   | geschlecht | name   | geschlecht | spezies |
+--------+------------+--------+------------+---------+
```

```
| Fluffy | w           | Claws  | m           | cat     |
| Buffy  | w           | Fang   | m           | dog     |
| Buffy  | w           | Bowser | m           | dog     |
+--------+-------------+--------+-------------+---------+
```

In dieser Abfrage werden zwei Aliase für den Tabellennamen angegeben, um es überhaupt zu ermöglichen, daß man sich mehrfach auf eine spezielle Spalte ein-und derselben Tabelle beziehen kann. Ohne die Aliase würde dies nicht funktionieren.






31.17 Wie man Informationen über Datenbanken und Tabellen erhält

Es könnte sein, daß man den Namen von Datenbanken oder einer Tabelle vergessen hat. Für diesen Fall gibt es in **MySQL** verschiedene Möglichkeiten, sich Informationen ausgeben zu lassen:

```
mysql> SELECT DATABASE();
+------------+
| DATABASE() |
+------------+
| menagerie  |
+------------+
```

Diese Ausgabe tritt immer dann auf, wenn man noch keine Datenbank ausgewählt hat, ein häufiger Fehler, wenn man gerade **MySQL** installiert hat, und sich erwartungsvoll an die Datenbank angekoppelt hat.

```
mysql> SHOW TABLES;
+---------------------+
| Tables in menagerie |
+---------------------+
| ereignis            |
| tier                |
+---------------------+
```

Aha, fein. Nu wollen wir explizit eine Beschreibung der Tabelle tier haben:

```
mysql> DESCRIBE tier;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| name     | varchar(20) | YES  |     | NULL    |       |
| besitzer | varchar(20) | YES  |     | NULL    |       |
| spezies  | varchar(20) | YES  |     | NULL    |       |
|geschlecht| char(1)     | YES  |     | NULL    |       |
| birth    | date        | YES  |     | NULL    |       |
| death    | date        | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
```

DESCRIBE beschreibt einige Dinge, über die wir uns bisher nicht gekümmert haben, z.B. Key, Null,

Default und Extra. Wir sehen die Tabelle nun in etwas anderer Form. Die gewohnten Spalten stehen in Zeilen, und dafür stehen die Eigenschaften in Spalten. Bei Type ist der Datentyp angegeben, Null zeigt an, ob die Spalte den Wert NULL enthalten darf, Key zeigt an, ob ein INDEX angelegt wurde, und Default zeigt an, welchen Wert die Zelle annimmt, wenn z.B. die Tabelle aus einer nicht ganz gefüllten ASCII - Datei geladen wurde.

Mit dem Feld INDEX hat es eine besondere Bewandnis, die die Geschwindigkeit der SQL-Abfragen dramatisch beschleunigen kann. Dazu mehr aber später.

Wenn Indizes in einer Tabelle angelegt wurden, oder wenn einen Spalte indiziert wurde, dann zeigt der Befehl:

```
SHOW INDEX FROM tbl_name produces
```

weitere Informationen an.








31.18 MySQL im Batch Modus

Wer bis hierhin fleißig war, der hat sicher viel eingetippt und ausprobiert. Insbesondere die CREATE TABLE Konstrukte können bei komplexen Datenbanken sehr lang sein. Wer einen Fehler gemacht hat, der muß alles nochmal eintippen. Schlecht - es geht besser. Man kann und sollte stets alle CREATE TABLE und SELECT Statements in eine Batchdatei, also eine ASCII-Datei schreiben. Editiert werden kann mit jedem beliebigen Editor. Insbesondere, wenn man sich noch in dem Anfangsstadium des Aufbaus einer Datenbank befindet, und noch viel nachbessern muß, macht sich eine Batchdatei schnell bezahlt. Wichtig ist es auch, SELECT Statements in eine Datei zu schreiben. Wie schon gesehen, sind einige Statements sehr lang und komplex. Wer mit großen Datenbanken hantiert (1-60 GByte) (ja, MySQL kann unter LINUX ALPHA, Solaris 2.7, Tru64 UNIX, und SGI IRIX so große Datenbanken handeln), der wird feststellen, daß die Durchläufe für monatliche Statistiken manchmal eine Nacht dauern. Die Ausgabe kann mitunter sehr groß sein. In diesem Fall läßt man die Ausgabe in eine Datei schreiben. Im Batch-Mode kann man diese Statements auch in die CRONTAB-Datei schreiben, mehr dazu nun im folgenden Abschnitt.

In allen vorangegangenen Abschnitten wurde **MySQL** interaktiv benutzt, um Abfragen zu starten. Die Ausgabe erfolgte jeweils auf den Bildschirm. Das Programm **mysql** ist das Frontend zum **mysqld**, nicht vergessen ! Ein Beispiel für den Batch-Modus:

```
shell> mysql < batch-file
```

Dieses Beispiel wird wohl auf Ihrer Datenbank nicht funktionieren, warum, das dürfte klarer werden, wenn man sich das nächste Beispiel anschaut:

```
shell> mysql -h host -u user -p < batch-file
Enter password: ********
```

Falls die Ausgabe zu lang ist, kann man sich die Seiten auch seitenweise anzeigen lassen:

```
shell> mysql < batch-file | more
```

Für die Ausgabe in eine Datei sollte man folgendes Beispiel testen:

```
shell> mysql < batch-file > mysql.out
```







32. Begriffsdefinitionen für Einsteiger







32.1 Attribut

Ein Attribut ist eine bestimmte Eigenschaft eiens Feldes in einer Tabellenspalte. Wie sich jedermann/frau vorstellen kann, werden Tabellenspalten durch die Angabe des Datentyps charakterisiert. Beispielsweise können diese Datentypen vom Typ DATE, CHAR, VARCHAR, INTEGER...u.s.w. sein. Attribute von Feldern charaktierisieren aber über den Datentyp hinaus das Verhalten der Datensätze in einer Spalte. Z.B. kann hier festgelegt werden, daß z.B. eine Spalte vom Typ VARCHAR(20) nicht mehr in der Größe wachsen darf, auch wenn jemand einen längeren String ablegen möchte. Das Attribut, daß das Feld variabel ist, ist meist die Standardeinstellung. Ein weiteres Attribut ist z.B. die Festlegung, daß ein Feld in einer Tabellenspalte konstant sein muß. Im Normalfall geht man davon aus, daß alle Werte verändert werden können, es gibt jedoch manchmal die Forderung nach der Festlegung von Werten, z.B. nach einem Monatsabschluß bei der Finanzzbuchhaltung oder bei einer Umstellung der MwSt. im April 1998 und zu Beginn 2000 ???. Ein weiteres Attribut kann sein, daß eine automatische Sortierung in auf-oder absteigender Reihenfolge vorgenommen wird. Alle Neueinträge werden automatisch einsortiert. Dies kann die Suche erheblich beschleunigen. Der Standardwert eines Feldes ist derjenige, der automatisch angenommen wird, wenn ein Feld gelöscht wird. Wer sich genauer über die Attribute von **MySQL** informieren möchte, der mag im Kapitel CREATE nachsehen. Wichtig ist hierbei zu wissen, daß nachträgliche Änderungen in Datenbanken schlecht möglich sind.







32.2 Primärschlüssel (primary key)

Der Primärschlüssel ist ein eindeutiger INDEX, also eine zusätzliche Spalte, die einem Datensatz einen eindeutigen Wert zuordnet. In einer großen Datenbank kann es stets passieren, daß in mindestens einer Spalte eines Datesatzes der Wert identsich ist. Bei Adressdatenbanken könnten z.B. mehrere Familien Schmitz in demselben Hochhaus wohnen. Namen und Adressen wären in der Tabelle gleich, wir hätten also einen doppelten Eintrag. Es müssen dann mehrere Spalten quasi als Primärschlüssel eingesetzt werden. Dieses nett man einen zusammengesetzen Schlüssel. Ein zusammengesetzer Schlüssel muß also immer soviele Spalten beinhalten, sodaß man einen Datensatz eindeutig bestimmen kann.







32.3 Fremdschlüssel (foreign key)

Ein Fremdschlüssel ist ein Feld einer Tabelle, welches auf den Primärschlüssel einer oder mehrerer weiteren Tabellen verweist. Beispielsweise könnte man zwei Tabellen anlegen. Eine mit den Einträgen Kundennummer, Firmenname, Anschrift und die andere (untergeordnete Tabelle) mit den Einträgen Kundennummer, Ansprechpartner und Telefonnummer. Die Kundennummer der ersten Tabelle ist dann als Primärschlüssel angelegt, die Kundennummer der zweiten Tabelle ist dann ein Fremdschlüssel. Es kann ja sein, daß z.B. bei einer großen Firma mehrere Ansprechpartner existieren. In diesem Fall wird also nur in der untergeordneten Tabelle ein weitere Eintrag angelegt, dessen Fremdschlüssel auf eine dieselbe Kundennummer, also auch auf denselben Firmennamen zeigt. MySQL unterstützt noch keine Fremdschlüssel. Daher muß man die SQL Statements so schreiben, daß mit INSERT und UPDATE alle Tabellen geupdatet werden.







32.4 INDEX

Ein INDEX ist meistens eine Spalte, nach der besonders häufig gesucht wird. Die Einträge in der Spalte werden als HASH Index gespeichert. Damit beschleunigt sich die Suche erheblich. Insbesondere bei Verknüpfungen von Tabellen wird MySQL dadurch erheblich schneller.







32.5 Relation (relation)

Eine Relation ist eine Verknüpfung zwischen zwei Tabellen. Diese kann mit Primär-und Fremdschlüssel hergestellt werden.

Zwei Beispiele:

DB1: CREATE TABLE adress (ident NUMERIC(10,0), name CHAR(30))

DB2: CREATE TABLE adress (ident INTEGER, NAME STRING)

Die Abfragesprache unter SQL ist ein solcher Standard, die sogenannte DML, Data Manipulation Language.




32.6 Aggregat Funktionen

1. **COUNT**
2. **SUM**
3. **AVG**
4. **MIN**
5. **MAX**

In einer Select-Anweisung können verschiedene Aggregatfunktionen verwendet werden.

COUNT

Diese Aggregatfunktion liefert die Anzahl Datensätze zurück, die mit der SQL-Anweisung eingeschränkt wurden. Beispiel:

Select Count(Ident) from Adressen

Das Ergebnis ist eine Zahl, die die Anzahl aller Ident bzw. aller Personen in der Tabelle Adressen darstellt.

SUM

Diese Funktion liefert die Summe aller Werte in der SQL-Ergebnismenge zurück. Beispiel:

Select sum(alter) from adressen

Das Ergebnis ist das Gesamtalter, die Summe aller Personen in der Tabelle Adressen

AVG

Mit dieser Funktion wird der Durchschnittswert zurückgegeben. Beispiel:

Select avg(alter) from adressen

Das Ergebnis besteht nur aus einem Wert, nämlich dem Durchschnittsalter aller Personen aus der Tabelle Adressen.

MIN

Es wird der kleinste Wert in der Datenmenge ermittelt und als Ergebnis zurückgegeben. Beispiel:

Select min(alter) from adressen

Das Ergebnis ist das jüngste Alter aller Personen in der Tabelle Adressen.

MAX

Es wird der grösste Wert in der Datenmenge ermittelt. Beispiel:

Select max(alter) from adressen

Das Ergebnis ist das grösste Alter aller Personen in der Tabelle Adressen.






32.7 Tupel

Als Tupel wird ein Datensatz, oder eine Zeile einer Tabelle bezeichnet. Der Ausdruck ist nur noch selten zu finden.







32.8 SQL DML (Data Definition Language)

Dies ist die Standardabfragesprache für relationale Datenbanken. Mit SQL DML ist fast alles auf Datenbankebene möglich um die Daten zu bearbeiten. Es können Daten selektiert, geändert und auch gelöscht werden. Da in vielen Endanwenderdatenbanken die genannten Möglichkeiten per Drag and Drop auf Tabellenebene anbieten, hat der normale Anwender SQL noch nie gesehen.







32.9 SQL DDL (Data Definition Language)

Dies ist die Standardsprache um relationale Datenbanken zu generieren. Mit SQL DDL ist es möglich die Objekte, wie Tabelle und Indizes zu erzeugen. Das Statement CREATE DATABASE gehört nicht zu dieser Sprache.







33. Installation von ODBC Treibern für Applikationen







33.1 ODBC Grundlagen (Open Database Connectivity)

ODBC ist eine Spezifikation von Microsoft© für ein Programminterface (API) welches Applikationen einen Zugriff auf verschiedenste SQL Datenbanken erlaubt. Wenn man ohne ODBC mit einer Applikation auf eine Datenbank zugreifen will, muß man explizit die Schnittstelle des Datenbankmanagementsystems ansprechen. Dieses sind datenbankspezifische DLLs. Wenn man den Datenbankhersteller für diese Applikation wechseln will, muss man ohne ODBC im Quellcode die Datenbankzugriffe auf die neue Schnittstelle der neuen Datenbank anpassen. Wenn eine Datenbank einen ODBC-Treiber mitliefert, und die Applikation via ODBC auf die Daten zugreift, so braucht der Quellcode der Applikation nicht geändert werden. Um wirklich datenbankunabhängig zu sein, sollte man nur den SQL-Standard in den Datenbankverbindungen benutzen. Die Data Definition Language (DDL), wie zum Beispiel Statements zum Erzeugen von Tabellen, sind datenbankspezifisch. Die Data Manipulation Language (DML) hingegen ist unter ANSI SQL (92 ...) hingegen weitestgehend vom Hersteller der Datenbank unanhängig. MySQL hat gewisse Abweichungen, während die freie Datenbank PostgreSQL sich an den ANSI SQL 92 Standard hält. Benutzt man die ODBC API (Application Programming Interface), so kann man gleichzeitig mehrere Datenbanken parallel mit einem (Windows 95) Client ansprechen. Hierzu genügt die unterschiedliche Benennung der Datenbank bei der Installation des zugehörigen ODBC-Treibers auf dem Client. Es gibt auch zahlreiche Nicht-SQL oder auch Nicht-Relationale Datenbanken mit einem ODBC-Treiber. Auch diese können mit eingebunden werden.







33.2 ODBC Treiberlevel

Weit verbreitet ist noch der ODBC 2.5 Standard, aktuell ist aber ODBC 3.0, welches zu ODBC 2.5 kompatibel ist, sofern man nur die zu der Version 2.5 gehörigen Funktionen verwendet. Hierzu erlauben alle ODBC Treiber eine Anpassung über das Konfigurationsmenü oder die Konfigurationsdatei.







33.3 ODBC Entwicklungs-Kits

Firma Microsoft hat für Windows Betriebssysteme eine ODBC 3.5 SDK entwickelt, mit welcher Hersteller von Datenbanken eigene ODBC-Treiber für Microsoft Applikationen entwickeln können. Voraussetzung hierzu ist, daß der Hersteller die Libraries an die Eigenheiten seiner (SQL) Datenbank anpasst, und diese auch frei zur Verfügung stellt.

Es gibt aber auch freie SDK's für UNIX, die unter GNU Public License zur Verfügung stehen, und z.B. die Entwicklung von ODBC-Treibern für Anwendungen unter UNIX erlauben (Staroffice, Framemaker, Applixware Office....). Hier gibt es zwei sich ergänzende Entwicklungen, IODBC (http://oasis.leo.org/perl/exts/db/ODBC/iodbs.dsc.html), und FreeODBC (http://users.ids.net/~bjepson/FreeODBC).

Auch für die freie Datenbank PostgreSQL gibt es ODBC-Treiber allerdings sind diese kommerziell. Siehe hierzu http://www.postgresql.org.

Desweiteren stellen verschiedene Firmen ODBC-Treiber her: OpenLink, SyBase, Simba Technologies, Visgenic Software, Intersolv, Concom Systems, die aber für MySQL nur wenig interessant sind.

Für **MySQL** gibt es FreeWARE Treiber für Microsoft Windows und UNIX auf der Homepage von MySQL. Diese entsprechen dem Standard 2.5, bieten allerdings keine Unterstützung für Cursor. Sie arbeiten aber hervorragend mit Microsoft Anwendungen zusammen. Für MySQL ist allerdings eine C API verfügbar, die Cursor für C-Programme zur Verfügung stellt. Auch unter PERL gibt es erfolgversprechende Ansätze für Cursor. Siehe auch das Kapitel Frontends und Cursor unter MySQL.







33.4 ODBC mit PERL

ODBC-Treiber können selbstverständlich mit verschiedensten Programmiersprachen angesprochen werden. Diese heißt unter Microsoft Windows **Win32::ODBC**. Die **::** stehen für eine Klassenbibliothek in PERL.

Unter http://www.bitmechanic.com/mail-archives/dbi-users/ findet man ein Mail-Archiv für das PERL DBI (DataBase Interface) mit unzähligen Beispielen und Tips.

Hier ein kleines Beispiel für das Ansprechen von MySQL mit PERL32.EXE und Win32::ODBC unter Microsoft Windows 95/98/NT:

```
use Win32::ODBC;

    Third you open a connection to your database with (note that this
example checks for failure):

        $DSN = "My DSN";
        if (!($db = new Win32::ODBC($DSN))){
            print "Error connecting to $DSN\n";
            print "Error: " . Win32::ODBC::Error() . "\n";
            exit;
        }

    Fourth you execute your SQL command (NOTE: due to backward compatibility
with NT::ODBC the Sql()
    method returns undef if it was successful and a non zero integer error
number if it fails):

        $SqlStatement = "SELECT * FROM Foo";
        if ($db->Sql($SqlStatement)){
            print "SQL failed.\n";
            print "Error: " . $db->Error() . "\n";
            $db->Close();
            exit;
        }

    Fifth you fetch the next row of data until there are no more left to
fetch. For each row you retrieve data and
    process it:

        while($db->FetchRow()){
            undef %Data;
            %Data = $db->DataHash();
            ...process the data...
        }

    Sixth you close the connection to the database:

        $db->Close();
```

Das Beispiel stammt von David Roth, der auch die PERL ODBC-Treiber für UNIX geschrieben hat, und der eine sehr lesenswerte ODBC-Homepage anbietet:

http://www.roth.net/perl/odbc/faq/. Bei irgendwelchen Problemen mit ODBC-Treibern sollte man sich unbedingt zuerst

hier umschauen.

Hier ein ähnliches Beispiel unter LINUX mit PERL:

```
#!/usr/bin/perl
use DBI();
$dbh=DBI->connect("DBI:mysql:database=test;host=host","user","password") or
print "$DBI::db_errstr\n";
```

Selbstverständlich können die Beispiele alle auch mit Embedded PERL, E-PERL u.s.w. zusammen mit dem APACHE-HTML-Server eingesetzt werden. Hierbei ändert sich die Schreibweise eventuell geringfügig.

Eine Referenz-Liste von Namhaften Unternehmen und Sites (Deutsche Bank, Siemens und www.deja.com) kann man auf http://perl.apache.org/embperl/Sites.pod.1.html#http_www_webpersonals_com_St einsehen.

Wer weitere Tips und Hinweise zu PERL such, der ist auf http://www.cgi-resources.com oder auf http://www.perl-archiv.de genau richtig.






34. MySQL und JAVA Frontends

JDBC (JAVA DataBase Connectivity) ist die Möglichkeit schlechthin, Frontends für SQL Datenbanken zu entwickeln, die auf allen Clients völlig unabhängig vom Betriebssystem laufen. Hierzu ist dann nur ein Browser erforderlich, der JAVA unterstützt. Hier kommen z.B. Netscape (Alle Betriebssysteme), der IE (nur Windows und Solaris), Opera (Windows, Mac) und sogar StarOffice 5.1 von SUN in Betracht. Selbstverständlich gibt es auch noch die Möglichkeit, das JDBC Frontend direkt von der JVM (JAVA Virtual Machine) auf der Arbeitsstation ausführen zu lassen. Ein guter Grund also, ich etwas eingehender mit JDBC zu beschäftigen.

Hier finden Sie z.B. eine vollständige SQL Datenbank mit JDBC Klasse 4 Interface: http://www.rent-a-database.de/mysql/mysqllive.html. Viel Spaß beim Ausprobieren der Demo !

In diesem Kapitel werden alle wichtigen Details zu JDBC erklärt, und einige Beispiele vorgestellt, in der Hoffnung, daß sich mehr Leute, die schon etwas Erfahrung mit C oder Basic haben, sich einfach einmal an JAVA und JDBC herantrauen. So schwer ist JAVA nun wirklich nicht.







34.1 Vor-und Nachteile von JDBC Treibern

Wie schon zuvor erwähnt, laufen JAVA Applets auf allen Betriebssystemen (write once - run everywhere). Leider gibt es Unterschiede bei JDBC Treibern, die verhindern, daß man von überall unproblematisch eine Verbindung zu einem SQL Server öffnen kann. Hier muß man zwischen den 4 verschiedenen Typen von JDBC Treibern unterscheiden:

- **Typ 1** ist eine sogenannte JDBC - ODBC Bridge, die einen ODBC Treiber auf der Arbeitsstation (Client) erfordert. Diese Lösung eigenet sich daher nicht für Clients im Internet, die mal eben auf Ihre MySQL Datenbank im Internet zugreifen möchten. Auch für Systemadministratoren bedeutet dies einen Mehraufwand, da auf jedem Client einen ODBC Treiber installiert werden muß. Zudem kommt noch hinzu, daß jedes Stück Software weitere Fehler beinhaltet und Probleme verursacht. Außerdem besitzt ODBC einen nicht ganz unerheblichen Protokoll - Overhead, sodaß man beim Zugriff über das Internet oder ISDN/Modem mit erheblichen Geschwindigkeitseinbußen rechen muß. Zusammenfassend muß man also sagen - völlig überholt.
- **Typ 2** ist eine herstellerspezifische Lösung, die anstelle der definierten ODBC Schnittstelle eine eigene, also proprietäre Schnittstelle verwendet. Diese muß dann natürlich sowohl auf dem Server als auch auf dem Client installiert sein. Die Kommunikation zwischen Client und Server findet also zwischen den beiden Treibern statt, und stellt ebenfalls eine Fehlerquelle dar. Sofern man nicht besondere Features dieser Schnittstelle benötigt, sollte man auf diese Variante verzichten.
- **Typ 3** stellt ebenfalls eine herstellerabhängige Schnittstelle dar. Serverseitig wird die Datenbank direkt über das Netzwerk (Port 3306/ 3333) angesprochen, auf der Seite des Clients exisitert dann eine JAVA-API, die auf dem Browser ausgeführt wird. Der Nachteil ist hier, daß diese JAVA Bibliotheken herstellerspezifisch sind. Falls das JAVA Applet über Browser gestartet wird, ist es zwingend erforderlich, daß Webserver und SQL Server auf derselben IP-Nummer erreichbar sind, da es ansonsten die Sicherheitsmechanismen im JAVA Browser ansprechen. Damit soll verhindert werden, daß der Browser Daten von z.B. der SQL Datenbank über eine Arbeitsstation wieder in das Internet verschicken kann. Damit könnten Einbrecher ansonsten Daten einfach aus dem Unternehmen ins Internet versenden, ohne daß der Anwender etwas davon merkt.
- **Typ 4** ist nun der Treiber, der nun einige dieser Nachteile nicht mehr besitzt. Schauen Sie sich die Live Demo einmal an: http://www.rent-a-database.de/mysql/mysqllive.html. WWW-Server und MySQL Server können irgendwo im Internet stehen und können auch lokal voneinander weit entfernt sein. Der WWW-Server liefert das JAVA Applet, welches in dem Browser gestartet wird. Der Browser baut so eigenständig eine Verbindung zu dem SQL Server auf. Der JDBC Typ 4 Treiber ist natürlich ebenfalls noch herstellerspezifisch, jedoch liefert SUN mit JAVA2 und den RMI Bibliotheken endlich auch Klassen mit, die es einem Programmierer erlauben, seine eigenen TCP/IP Protokolle zum Zugriff auf die SQL Datenbank zu schreiben. Man kann hier recht einfach vorhandene Klassen für seine SQL Datenbank anpassen. Für MySQL gibt es gleich 2 verschiedene JDBC Typ 4 Treiber, die auch hervorragend funktionieren. Dieser JDBC Typ 4 Treiber ist als der einzige, der für MySQL Datenbanken im

Internet ohne Probleme seitens der Clients und des Servers verwendet werden kann.





34.2 Cursor unter JDBC

SUN´s RMI JDBC Bibliotheken sind sicher die Bibel für Programierer, die man zuerst lesen sollte. Leider unterstützt SUN nur die sogenannten Client Cursor in der **forward cursor** Variante, was bedeutet, daß das **result set** mit **SELECT .. FROM ...LIMIT 1,20..** seitenweise abgerufen und dargestellt wird. Ein Scrolling zurück ist damit nicht möglich, es wird aber auch wenig benötigt. Es gibt allerdings Hersteller, die Klassenbibliotheken auch hierfür anbieten.

Verwendet man nun die von SUN implementierte JDBC-ODBC Bridge für MySQL, dann können **forward cursor** dank der DBLibrary von Microsoft ebenfalls verwendet werden. Diese ähneln vom Aufruf her den Serverside Cursorn, auch wenn diese nur von der DBLibrary emuliert werden. MySQL liefert nur die Daten, mehr nicht.

Hier nun ein paar Beispiele, wie man allgemein diese emulierten Cursor unter JAVA anspricht: http://www.rent-a-database.de/mysql/mysqllive.html

Und nun ein paar Ausschnitte des Quellcodes:

```
Statement stmt = _connection.createStatement();
stmt.setCursorName("author_cursor");
ResultSet rs = stmt.executeQuery("SELECT au_id,
     au_lname, au_fname FROM authors
     WHERE city = 'Oakland'");
while(rs.next())
{
   String id =  rs.getString(1));
.....
.....
Statement stmt1 = _connection.createStatement();
Statement stmt2 = _connection.createStatement();
stmt1.setCursorName("author_cursor");
ResultSet rs = stmt1.executeQuery("SELECT
         au_id,au_lname, au_fname
         FROM authors WHERE city = 'Oakland'
         FOR UPDATE OF au_lname");
String cursor = rs.getCursorName();
String last_name = new String("Smith");
while(rs.next())
{
   if (rs.getString(1).equals("274-80-9391")
    {
      stmt2.executeUpdate("UPDATE authors
            SET au_lname = '"last_name + "'
```

```
                WHERE CURRENT OF " + cursor);
    ...
    ...
    stmt2.executeUpdate("DELETE FROM authors
             WHERE CURRENT OF " + cursor);
```

Mehrfache Cursor sind unter JAVA ebenfalls möglich. Hier ein Beispiel:

```
    Statement stmt1 = con.createStatement();
    stmt1.setCursorName("authCursor");
    ResultSet rs1 = stmt1.executeQuery("select * from authors");
    while(rs1.next())
    {
        // ... do whatever with the results until...
        // if (some condition w/regards to rs1)
        Statement stmt2 = con.createStatement();
        stmt2.setCursorName("titleCursor");
        ResultSet rs2 = stmt2.executeQuery(
"select * from titles for READ ONLY");
        // we want this readonly cursor to be efficient, so...
        Statement stmt3 = con.createStatement();
        stmt3.executeUpdate("set cursor rows 100 for titleCursor");
        while (rs2.next())
        {
           // on the first next() call, the server sends back the
           // first 100 rows of the titles table.  on subsequent
           // rs2.next() calls, we already have the row waiting
           // on the network to be read -- we do not have to do
           // another round-trip to the database to fetch the next
           // row.
    ....
    ....
```

Es ist hierbei völlig egal, welche SQL Datenbank zum Einsatz kommt, da die SQL Datenbanken von der Cursorpositionierung ja nicht betroffen sind (clientsside cursor). Hier sind es die Clients, die an Stelle des SQL - Servers die Aufgaben des Pre-und Postprozessors übernehmen. Die SQL Datenbank liefert auf die Anforderung eines SELECT Statements nur die Daten, alles weitere übernehmen die intelligenten Clients.






34.3 Bibliotheken für JAVA (JDBC) Frontends

Zum Schreiben von JAVA Frontends benötigt man auf der einen Seite JDBC Treiber für die Datenbankanbindung, und auf der anderen Seite Klassenbibliotheken für die GUI (Graphical User Interface), also die Buttons, u.s.w. Nun ist es so, daß sich unter JAVA Bibliotheken eigentlich ohne Probleme verwenden lassen müßten. Ein Problem ist hierbei, daß auch JAVA sich weiterentwickelt hat, von JAVA 1.0.x zu 1.2.x, auch JAVA 2.0 genannt. Da die Browser ja den JAVA Code ausführen können müssen, ist es erfordlich, daß man Bibliotheken verwendet, die die Browser unterstützen.

Wir wünschen uns natürlich, daß alle Browser, also auch die etwas älteren, die JAVA bereits unterstützen, unser JAVA Applet anstandslos ausführen, eine Verbindung zu der SQL Datenbank aufbauen, und die Ergebnisse nach allen Regeln der Kunst auch hübsch darstellen.

Bezüglich der GUI muß man feststellen, daß die sog. SWING Bibliothek im Moment das NONPLUSULTRA darstellt, was man so an Benutzerkomfort erwarten kann. Hier kann man das LOOK-AND-FEEL sogar selber bestimmen. SWING wird von SUN immer mitgeliefert. Natürlich gibt es auch Klassen von anderen Herstellern, die auch ganz nette Features besitzen, Standard sind diese jedoch nicht.

Ältere Browser müssen also diese SWING Bibliotheken unterstützen können. Hierzu gibt es eine einfache Möglichkeit, diese durch den Browser vom Server nachladen zu lassen:

```
<HTML>
        <HEAD>
                <TITLE>MySQL JDBC Applet

        <BODY><H1>Die Bibliotheken werden geladen, bitte etwas Geduld

        <APPLET ARCHIVE="mysql.jar, swingall.jar" CODE="mysql.class" WIDTH="720"
                HEIGHT="350">
```

Hiermit können auch ältere Browser, die das JDK 1.1 unterstützen, die SWING-Bibliotheken für die GUI nachladen. Diese werden auf der Festplatte der Clients abgelegt, und beim nochmaligen Zugriff dann direkt ohne neues Nachladen angesprochen. Die Bilbiothek **mysql.jar** enthält Ihre Klassenerweiterungen für Ihr Applet mit allen spezifischen Details Ihrer MySQL Datenbank (IP-Nummer, Port, Tabellen, Spalten ....).

Bezüglich der JDBC Treiberklasse haben wir nun gleich zwei JDBC Typ 4 Treiber zur Verfügung: > und http://doc.trader.dk/mysql/twz-jdbc/jdbc/mysql/docs/twz1jdbcForMysql-doc.html . Der letztere Treiber ist der neuere Typ 4 Treiber, der auch von den Entwicklern von MySQL empfohlen wird.

Wir müssen diese beiden Klassenbibliotheken nun verwenden, um unser Frontend für MySQL zu schreiben. Laden Sie sich hierzu diese Bilbiotheken auf Ihren LINUX Rechner.







34.4 Kompilieren und Start eines Beispiels

Als JAVA Kompiler kann man jeden JAVA Kompiler verwenden. Hierzu bietet sich natürlich der auf der Homepage von SUN kostenlos angebotene Kompiler an. Alternativ findet man auf der Homepage von IBM den IBM eigenen JAVA-Kompiler. Auch unter LINUX können Sie auch den freien JAVA Kompiler GUAVE verwenden. Alle erzeugen aus dem JAVA Quellcode ein JAVA Code-Binary, welches die Browser dann ausführen. Unser Beispiel bezieht sich auf nun auf LINUX mit installiertem JAVA-Kompiler von SUN, der auf fast allen LINUX Distributionen enthalten ist.

Das Problem ist hierbei, daß man die Kompiler dazu bewegen muß, die Klassenbibliotheken, die im Header des MySQL Applets verwendet werden sollen, auch tatsachlich einzubinden. Hierzu muß man zuerst alle Bibliotheken, also SWING, JDBC u.s.w. auf der Festplatte in einem Verzeichnis liegen haben. Danach muß man die Umgebungsvariable **CLASSPATH** darüber unterrichten, wo die SWING Bibliothek zu finden ist (export CLASSPATH="./swingall.jar" unter LINUX). Nun entpacken Sie alle JAVA - Klassen und das mysql Applet **mysql.java** in ein Verzeichnis. Übersetzen Sie nun den Quellcode mit **javac mysql.java**. Danach können Sie alle fertig kompilierten Klassen in dem Archiv **mysql.jar** zusammenfassen: **jar cvf adressen.jar**. Wir haben nun u.a. folgende Dateien in unsererm Verzeichnis:

```
mysql.class
mysql.jar
swingall.jar
mysql.html
```

Diese Dateien können Sie nun einfach auf Ihren Webserver kopieren, und im Browser die Datei **mysql.html** aufrufen. Die HTML Datei wird gestartet, wobei die Bibliotheken mysql.jar und swingall.jar zuerst auf die Arbeitsstation geladen werden. Danach baut sich die Verbindung zu MySQL Server auf und sie können nun die Daten anzeigen und editieren.

Die Quellcodes zu unserem Beispiel **mysql** finden Sie unter http://www.rent-a-database.de/support/beispiele/jdbc/.







34.5 Datentypen - Konvertierung MySQL<->JAVA

In Arbeit !







34.6 JAVA Bibliotheken zum Zugriff auf SQL Datenbanken

Es gibt inzwischen zahlreiche Hersteller von JAVA Bibliotheken, die alle auf Klasse 4 JDBC Treibern aufsetzen. Eines der hervorragenden Beispielen ist die **SQLJ** Bibliothek, die es erlaubt, die SQL Statementsin den JAVA Quellcode zu schreiben. Diese werden dann in entsprechende JDBC Anweisungen übersetzt. (In Arbeit!)







34.7 Links und Beispiele

Hier ein Class 4 JDBC-Treiber mit Beispielen: http://www.voicenet.com/~zellert/tjFM/, und der entsprechenden Dokumentation dazu: http://www.voicenet.com/~zellert/tjFM/twz1jdbcForMysql-doc.html Klasse 4 JDBC-Treiber kommunizieren ohne Zwischenschicht direkt über JAVA Sockets mit der Datenbank und sind deshalb Plattformunabhängig, lassen sich also mit MAC, Microsoft, LINUX, SUN ....Betriebssystemen starten.

http://doc.trader.dk/mysql/twz-jdbc/jdbc/mysql/docs/twz1jdbcForMysql-doc.html Klasse 3 JDBC-Treiber ist z.B. der JDBC-ODBC-Treiber mit der RMI JAVA Klasse von SUN. Klasse 2 JDBC Treiber, wie z.B. JDBC OCI (Oracle Call Interface) erfordern Treiber auf der Client-Seite und auf der Serverseite.ier noch ein etwas veralteterJDBC Treiber für MySQL....von GWE Technologies.... http://www.gwe.co.uk/java/jdbc/

Ein interessantes Beispiel ist TableGen, ein Tablegenerator für SQL-Datenbanken mit JDBC Klasse 4 Treiber. http://freespace.virgin.net/joe.carter/TableGen/index.html

Brent's JAVA Page ist für Leidensgenossen gedacht, die Beispiele für JAVA und SQL-DAtenbanken suchen. http://www.ewin.org/~bret/java/ Hier ist für fast alle Datenbanken etwas dabei !

Für Entwickler ganz interessant sein dürfte der GIANT JAVA TREE, siehe http://www.gjt.org/doc/, der einen komfortablen Class-Browser (Symantec...) ersetzen kann..... ICE.COM bietet für SQL eine ganze Reihe von Applets mit Quellcode als GPL Freeware an. Hier zeigt sich, was man alles mit JDBC-Treibern und JAVA praktischmachen kann. Hier ein Screenshot es Mail Clients unter Windows NT 4: http://www.ice.com/java/icemail/screen1.html http://www.ice.com/java/sqlclient/

Darüber hinaus gibt es Treiber mit EXTENSIONS, wie z.B. Support für CURSOR, Prefetching von Daten, Caching von Daten u.s.w., die alle die Performance des Datenbank erheblich verbessern können (in der Praxis sieht das schlechter aus). JDBC-ODBC-Treiber sind auf der Site von Pierre Gibello auf http://dyade.inrialpes.fr/mediation/download/RmiJdbc/RmiJdbc.html angeboten. Dieser JDBC Treiber ist mit JAVA RMI von SUN erstellt worden. Siehe hierzu http://java.sun.com/products/jdbc/faq.html Wer sich ausführlich über JDBC-Treiber informieren möchte, der kann auf http://www.retep.org.uk/postgres/ bei dem Entwickler des PostgreSQL JDBC-Treiber, Peter T. Mount umschauen.

Die Site von Ken North ist ebenfalls sehr lesenswert: http://ourworld.compuserve.com/homepages/Ken_North/JDBCVend.htm.

Für Entwickler ist die original SUN JDBC Site vielleicht interessant: http://java.sun.com/products/jdbc/

Hier ein einfaches Beispiel für ein Klasse 3 RMI JDBC-Treiber, der mit allen SQL-Datenbanken, also auch mit **MySQL** funktioniert. So einfach kann das Leben sein !

```
import java.sql.*;
import java.net.InetAddress;
/**
 * This is a sample program for RmiJdbc client/server jdbc Driver
 * RmiJdbc relies on Java RMI for jdbc objects distribution
 */
public class rjdemo {

  public static void main(String[] args) {
    try {

      // Register RmiJdbc Driver in jdbc DriverManager
      // On some platforms with some java VMs, newInstance() is necessary...
```

```
        Class.forName("RmiJdbc.RJDriver").newInstance();

        // Test with MS Access database (rjdemo ODBC data source)
        String url = "jdbc:odbc:rjdemo";

        // RMI host will point to local host
        String rmiHost = new String(
         "//" + InetAddress.getLocalHost().getHostName());

        // RmiJdbc URL is of the form:
        // jdbc:rmi://<rmiHostName[:port]>/<jdbc-url>

        java.sql.Connection c = DriverManager.getConnection("jdbc:rmi:"
         + rmiHost + "/" + url, "admin", "rjdemo");

        java.sql.Statement st = c.createStatement();
        java.sql.ResultSet rs = st.executeQuery("select * from contact");

        java.sql.ResultSetMetaData md = rs.getMetaData();
        while(rs.next()) {
          System.out.print("\nTUPLE: | ");
          for(int i=1; i<= md.getColumnCount(); i++) {
            System.out.print(rs.getString(i) + " | ");
          }
        }

        rs.close();

      } catch(Exception e) {
        e.printStackTrace();
      }
    }
  };
```

Von George Reese (Imaginary) werden für mSQL, also indirekt auch für **MySQL** JDBC-Applets mit Quellcodes angeboten, reinschauen lohnt sich auf jeden Fall: http://www.imaginary.com/Java/

Von ihm stammt dieses Gästebuch in JAVA mit mSQL Datenbank auf der Site: http://www.imaginary.com/~borg/GuestBook/GuestBook.html. Hier ein Beispiel von George Reese für die mSQL Anbindung:

```
import java.sql.*;
public class Select {
    static public void main(String[] args) {
        try {
            String url;
            Connection conn;
            Statement stmt;
            ResultSet rslt;

            url = "jdbc:msql://carthage.imaginary.com:1114/test";
            conn = DriverManager.getConnection(url, "me", "");
            stmt = conn.createStatement();
            rslt = stmt.executeQuery("SELECT * FROM test");
            while( rslt.next() ) {
                System.out.println("ID: " +
                                   rslt.getInt(1) + "\n"
                                   "Value: " +
```

```
                                    rslt.getString(2));
            }
            conn.close();
        }
        catch( Exception e ) {
        }
    }
}
```

Dasselbe Beispiel für SYBASE ASE Datenbank unter LINUX. SYBASE für LINUX ist ein kleiner Geheimtip, da SYBASE voll Microsoft SQL kompatibel ist, und auch kostenlos ausgetestet werden kann. Dieses Beispiel ist gegenüber dem oberen ein wenig ergänzt worden, die Parallelen sollten aber deutlich werden...:

```
/**
 * Generic connection engine,
 */
import java.sql.Connection;
import java.sql.Date;
import java.sql.SQLException;
import java.sql.Statement;
import java.sql.ResultSet;
import java.sql.DriverManager;


public class ConnectExapmple {

        private Connection con = null;
        private Statement s = null;

        private Connection connect()
        {
                String dbdriver = "com.sybase.jdbc.SybDriver";
                String dburl = "jdbc:sybase:Tds:10.200.38.232:7100";
                String dbuser = "sa";
                String dbpassword = "";

                try {
                        Class.forName(dbdriver);
                        return DriverManager.getConnection (dburl, dbuser,
dbpassword);
                }
                catch (Exception e) {
                        System.out.println("Can't connect");
                        e.printStackTrace();
                        return null;
                }
        }

        public ResultSet execute(String query)
        {

                ResultSet       r;

                try {
                        s = con.createStatement();
                        r = s.executeQuery(query);
                        return r;
                }
```

```
                catch (Exception e) {
                        e.printStackTrace();
                        System.exit(1);
                }
                return null;
        }

        public void test() {

                ResultSet r;
                String theQuery = new String("select name from sysobjects
where type='U'");

                try {
                        con = connect();
                        r = execute(theQuery);
                        while (r.next()) {
                                System.out.println(r.getString("name"));
                        }
                        con.close();
                }
                catch (Exception e) {
                        e.printStackTrace();
                }
        }

        public static void main(String args[])
        {
                ConnectExample db = new ConnectExample();

                db.test();
        }
}
```

Damit diese Beispiele auch laufen, muß man den SUN JAVA-Interpreter und die Klassenbibliotheken auf der Arbeitsstation installiert haben. Im Browser funktionieren die Beispiele auch, siehe das Gästebuch. Wenn man sich die Quellcodes von dem JDBC-Treiber für mSQL genau anschaut, dann fallen einige Unterschiede auf, die sich teilweise recht einfach kompensieren lassen. Dazu gehört z.B. die Standard Portnummer. Andere, wie z.B. die Unterstützung für Hersteller - spezifische Variablentypen müssen im Quellcode des JDBC-Treibers erst angepasst oder implementiert werden. Viele der Routinen, die SUN mit der RMI zur Verfügung stellt, werden von mSQL nicht unterstützt, diese sind nur als **void** (leer) Routinen implementiert, funktionieren also nicht. Beim Aufruf dieser Routinen wird ein Fehler ausgegeben.

Brian Jepson hat auf seiner Homepage http://users.ids.net/~bjepson/javadb/ zahlreiche Programme zu seinem Buch, **JAVA Database Programming** veröffentlicht, die zahlreiche Beispiele zur JAVA/JDBC-ODBC-Bridge-Programmierung und SQL Datenbanken enthalten. Darryl Collins hat auf seiner Homepage http://mama.minmet.uq.oz.au/msqljava/ ein paar sehr schöne Anleitungen zum Progammieren eines JAVA-Interface für SQL-Datenbanken (mSQL und MySQL) veröffentlicht, begleitet von ein paar funktionierenden Online-Demos. Ein gebunden wird das JAVA-Applet einfach an einer beliebigen Stelle im HTML-Code:

<applet code="Demo.class" width=400 height=200> </applet>

Dieses Applet wird einfach mit dem SUN JAVAC Compiler kompiliert - fertig !

```
import java.awt.*;
import java.lang.*;
import java.util.*;
```

```
import msql.*;

public class Demo extends java.applet.Applet {

        TableView view;

        public void init() {

                Msql msql;
                MsqlResult date;

                try {
                        msql = new Msql();
                        msql.Connect("mama.minmet.uq.oz.au","nobody");
                        msql.SelectDB("demo");

                        date = msql.Query(
                                "select * from domain order by Count desc"
                        );

                        msql.Close();

                        view = new TableView(date);
                        add(view);

                } catch (MsqlException e) {
                        System.out.println(e.getMessage());
                }
        }
}


class TableView extends Panel {

        RowView label;
        String names[];

        RowView row[];
        MsqlResult result;

        Scrollbar rowbar;

        int nCursor;

        TableView(MsqlResult result) {

                int i;

                this.result = result;

                int f = result.NumFields();
                names = new String[f];

                for(i=0; i<f; i++) {
                        String s[] = result._FetchField();
                        names[i] = new String(s[1]);
```

```
                }

                label = new RowView(names);

                int n = result.NumRows();
                row = new RowView[5];

                Panel table = new Panel();
                table.setLayout(new GridLayout(5,1,0,0));

                nCursor = 0;

                result.DataSeek(nCursor);
                for(i=0; i<5; i++) {
                        row[i] = new RowView(result.FetchRow());
                        table.add(row[i]);
                }

                rowbar = new Scrollbar();
                rowbar.setValues(0,5,0,n-5);

                setLayout(new BorderLayout());
                add("North",label);
                add("Center",table);
                add("East",rowbar);

        }

        public boolean handleEvent(Event e) {

                int i;

                if(e.target instanceof Scrollbar) {

                        if(e.target == rowbar) {
                                nCursor = rowbar.getValue();

                                result.DataSeek(nCursor);
                                for(i=0; i<5; i++)
                                        row[i].setText(result.FetchRow());

                                return true;
                        }
                }

                return false;
        }
}

class RowView extends Panel {

        TextField field[];

        RowView(String s[]) {

                setLayout(new FlowLayout(FlowLayout.LEFT,0,0));
```

```
                field = new TextField[s.length];
                int width[] = new int[s.length];

                width[0] = 32;
                width[1] = 8;


                for(int i=0; i<s.length; i++) {
                        field[i] = new TextField(s[i],width[i]);
                        field[i].setEditable(false);
                        add(field[i]);
                }
        }

        public void setText(String s[]) {

                for(int i=0; i<s.length; i++)
                        field[i].setText(s[i]);
        }
}
```

Wer sich dieses Beispiel live im Internet anschaut, der wird bemerken, daß dieses Beispiel sehr ausbaufähig ist. **Cursor**, die man schmerzlich bei mSQL und MySQL vermißt, werden hierdurch weitestgehend überflüssig, da das JAVA-Applet Ergebnisse cachen kann. Da hierdurch beim Scrollen keine Serverlast entsteht, kann auch mit vielen Clients gleichzeitig auf mSQL oder MySQL zugegriffen werden. Allerdings bietet die SWING-Klassenbibliothek wesentlich mehr Programmierkomfort, als die älteren AWT-Klassen.








35. Anbindung von ACCESS an MySQL

MySQL besitzt einen ODBC Treiber, der es ermöglichen, alle Microsoft Programme oder auch StarOffice 5 oder Applixware Office an MySQL anzubinden. Im Falle von ACCESS ergeben sich hier einige Vorteile:

- Geschwindigkeitsverbesserungen zwischen Faktor 3 und 100
- Entlastung des LAN´s
- Zentrale Lagerung und Backups der Inhalte
- Abruf der Daten über ein WWW-Interface (PHP3, PERL, ASP)
- Anbindung einer SHOP-Lösung (INTERSHOP, LITTLE-IDIOT-SHOP, MINIVEND)
- Kombination mehrerer SQL Datenbanken miteinander
- Dezentrale Erfassung von Daten über den Replikationsmechanismus von ACCESS
- Verteilte Datenbanken über Filialen hinweg durch Replikations - Frontends unter MySQL

Wie Sie sehen, lassen sich über ODBC fast alle Anwendungen unter Windows an MySQL anbinden. MySQL läuft netterweise auch unter NT, und das absolut stabil. Zur Installation schauen Sie sich bitte das Tutorial ODBC und die Kapitel zur Installation Installation von MySQL unter Windows an.




36. Tutorial Testverbindung von PHP3 und MySQL

Ziel dieses Tutorials ist es, eine beliebige Datenbank im Internet oder im Intranet mit einem allgemeingültigen PHP3 Skript anzusprechen, und uns alle Daten einer Tabelle ausgeben zu lassen. Es ist hierbei auch völlig egal, wo das PHP3 Skript läuft. Unser Beispiel hier bezieht sich auf eine im Intranet unter der IP-Nummer 10.0.0.5 erreichbaren LINUX Server mit MySQL und einer Arbeitsstation, auf der Sie nach unserer Anleitung den MySQL und Apache Server mit PHP3 installiert haben. Siehe zur Installation auch das Kapitel WAMP (Windows, Apache, MySQL, PHP3).

Starten Sie MySQL und Ihren Apache Server unter Windows 98. Klicken Sie hierzu auf **[START]->[PROGRAMME]->[Apache Web Server]->[Start Apache as console app]** und starten Sie das Binary **mysqld.exe** im Verzeichnis **c:\mysql\bin\mysqld.exe**, falls Sie MySQL auf Ihrer Arbeitsstation installiert haben.

Kopieren Sie die Datei in das Verzeichnis **c:\apache\htdocs**

```
<html>
<?php
  // Verbindung zur Datenbank auf 10.0.0.5 als
  // Alternativ können Sie auch www.rent-a-database.de einsetzen.
  // User testuser ohne Paßwort.
  $link = mysql_connect("10.0.0.5", "testuser", "testpasswort");

  // Auswahl der zu verwendenden Datenbank auf dem Server
  $query = "use test";
  if (!mysql_query($query, $link))
    die("Datenbank test existiert nicht.<br>\n");

  // Auslesen der Tabelle auth_users in dieser Datenbank
  $query = "select * from testtabelle";
  $res = mysql_query($query, $link);
  if (!$res)
    die("Anfrage $query scheitert.<br>\n");

  // Bestimme Größe des Ergebnisses
  $rows = mysql_num_rows($res);
  $cols = mysql_num_fields($res);
  printf("Anfrage ergibt %d Zeilen zu %d Spalten.<br>\n", $rows, $cols);

  // Durchlesen des Ergebnisses
  while($d = mysql_fetch_array($res)) {

    // $d ist ein Array aus Spaltenname, aktuellem Wert
    reset($d);
    while(list($k, $v) = each($d)) {
      printf("%s = %s, ", $k, $v);
    }
    printf("<br>\n");
  }
?>
```

Hierzu öffnen Sie mit [START]->[Programme]->[Zubehör]->[Editor] den Windows Editor und Fügen mit Hilfe der Zwischenablage Cut - und - Paste Funktion die Datei aus dem Browser in Ihren Editor ein. Mit **Datei Speichern unter** speichern Sie diese Datei im Verzeichnis **c:\apache\htdocs** als Datei **test.php3** ab.

test.php3 - Editor

Datei Bearbeiten Suchen ?

```
<html>
<?php
  // Verbindung zur Datenbank auf localhost als
  // Alternativ können Sie auch www.rent-a-databse.de einsetzen.
  // User testuser ohne Paßwort.
  $link = mysql_connect("10.0.0.5", "testuser", "testpasswort");

  // Auswahl der zu verwendenden Datenbank auf dem Server
  $query = "use test";
  if (!mysql_query($query, $link))
    die("Datenbank test existiert nicht.<br>\n");

  // Auslesen der Tabelle auth_users in dieser Datenbank
  $query = "select * from testtabelle";
  $res = mysql_query($query, $link);
  if (!$res)
    die("Anfrage $query scheitert.<br>\n");

  // Bestimme Größe des Ergebnisses
  $rows = mysql_num_rows($res);
  $cols = mysql_num_fields($res);
  printf("Anfrage ergibt %d Zeilen zu %d Spalten.<br>\n", $rows, $cols);
```

Abbildung:Windows Editor

Starten Sie nun Ihren Browser und geben Sie als URL **http://localhost/test.php3** ein. Was Sie nun sehen, ist die Ausgabe der Anfrage des PHP3 Skriptes an die MySQL Datenbank. Wir sind im Prinzip nun am Ziel angelangt. Wir haben eine MySQL Datenbank mit einem HTML Interface installiert:

http://localhost/test.php3 - Microsoft Internet Explorer

Datei Bearbeiten Ansicht Favoriten Extras ?

Zurück Vorwärts Abbrechen Aktualisier... Startseite Suchen Favoriten Verlauf E-Mail

Adresse http://localhost/test.php3 Wechseln zu

Links Channel Guide Das Beste im Web Internetstart Kostenlose HotMail Links anpassen Microsoft

Anfrage ergibt 7 Zeilen zu 2 Spalten.
0 = 5, spalte1 = 5, 1 = test, spalte2 = test,
0 = 5, spalte1 = 5, 1 = testwert, spalte2 = testwert,
0 = 34567, spalte1 = 34567, 1 = kannix und istnix weissnix habenix, spalte2 = kannix und istnix weissnix habenix,
0 = 5, spalte1 = 5, 1 = test, spalte2 = test,
0 = 5, spalte1 = 5, 1 = testwert, spalte2 = testwert,
0 = 34567, spalte1 = 34567, 1 = kannix und istnix weissnix habenix, spalte2 = kannix und istnix weissnix habenix,
0 = 43, spalte1 = 43, 1 = fdsfdsfs, spalte2 = fdsfdsfs,



Abbildung:Browser Ausgabe: testtabelle

Schauen wir uns noch einmal genauer die Ausgabe an:

```
Anfrage ergibt 7 Zeilen zu 2 Spalten.
0 = 5, spalte1 = 5, 1 = test, spalte2 = test,
0 = 5, spalte1 = 5, 1 = testwert, spalte2 = testwert,
0 = 34567, spalte1 = 34567, 1 = kannix und istnix weissnix habenix, spalte2
= kannix und istnix weissnix habenix,
0 = 5, spalte1 = 5, 1 = test, spalte2 = test,
0 = 5, spalte1 = 5, 1 = testwert, spalte2 = testwert,
0 = 34567, spalte1 = 34567, 1 = kannix und istnix weissnix habenix, spalte2
= kannix und istnix weissnix habenix,
0 = 43, spalte1 = 43, 1 = fdsfdsfs, spalte2 = fdsfdsfs,
```

Wir haben nun die Tabelle **testtabelle** so ausgegeben, daß unser PHP3 Skript über alle Spalten und Zeilen die Tabelle abgesucht hat, und die Ausgabe einfach nur gedruckt hat. Dieses PHP3 Skript ist natürlich sehr praktisch, weil man es auf alle Tabellen anwenden kann. Man muß nur oben den Server angeben, Login und Paßwort eintragen, die Datenbank und Tabelle angeben und schon werden einem alle Daten aus dieser Tabelle über den Browser angezeigt. Ändern kann man natürlich hier noch nichts. Wir möchten ja auch nur sehen, ob die Ansteuerung der Datenbank ja auch korrekt funktioniert. Verwenden Sie dieses **test.php3** Skript stets dazu, um zu testen, ob Ihre Datenbank auch ansprechbar ist.

Vergleichen Sie obige Ausgabe einmal mit der Ausgabe des MySQL Administrator für Windows:

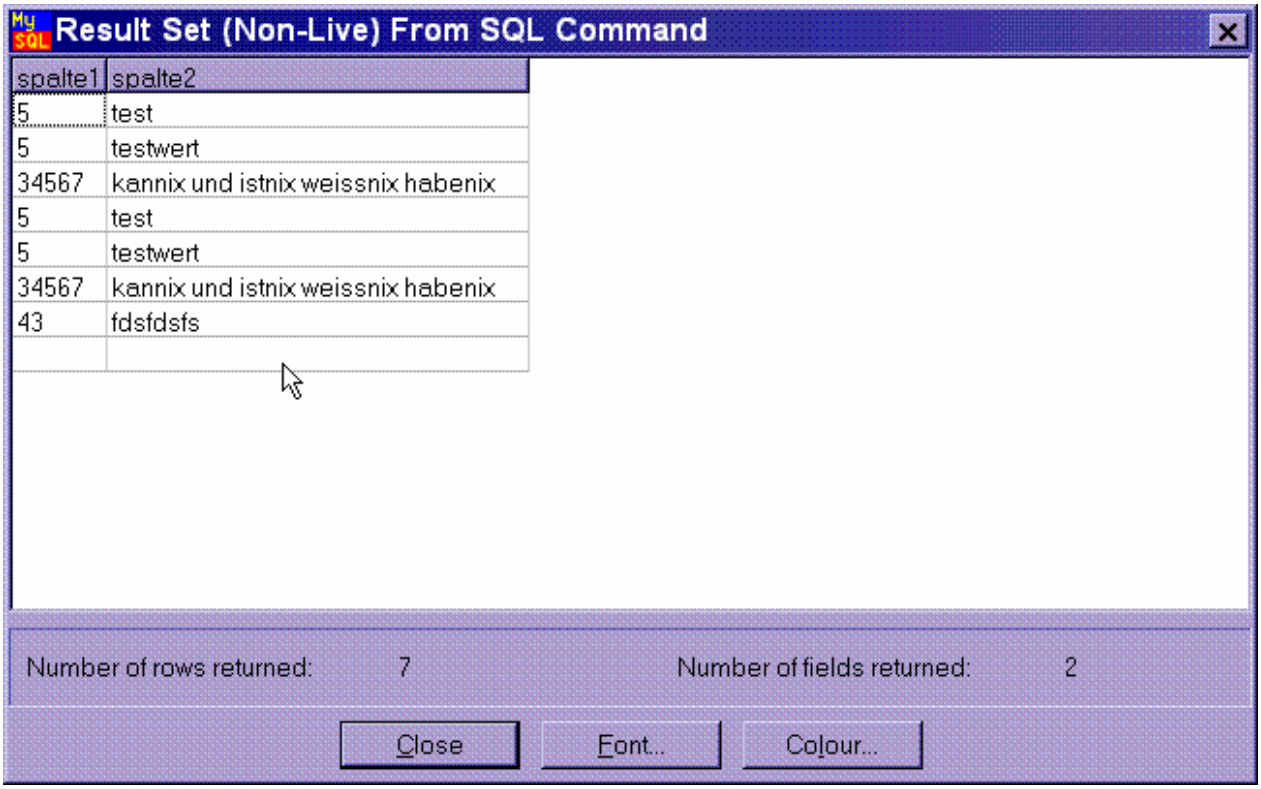


Abbildung:MySQLAdmin: testtabelle

Sie können nun ein wenig im PHP3 Skript "herumspielen" und sich die Ausgabe mit ein paar HTML Tags noch verschönern. Dieses Sktipt **test.php3** können Sie stets als Ausgangsbasis für die Ausgabe von Daten aus der Datenbank verwenden. Wir wünschen nun viel Spaß !

Sie können aber auch einmal unser Live Beispiel ausprobieren ! Hierzu müssen Sie JAVA in Ihrem Browser aktiviert haben. Klicken Sie nun auf http://www.rent-a-database.de/mysql/mysqllive.html und probieren Sie einige Beispiele aus.






37. Tutorial für PHP 3.0/4.0 WWW-Interfaces







37.1 Vorbereitung

Für den Anfang reicht im Prinzip dieses Tutorial völlig aus. Wer sich an LINUX herantraut (es ist recht einfach zu installieren), der ist sicher mit der S.u.S.E. 6.0/6.1 Distribution gut beraten. S.u.S.E. hat nämlich auf die Lauffähigkeit von **MySQL** und dem APACHE 1.3.x Server mit den Modulen für PERL und PHP3 viel Wert gelegt. Wer diese Pakete installiert, der kann davon ausgehen, daß diese auch bei Start des Servers aktiv sind. Wichtig hierbei ist es zu wissen, daß man die PHP Interpreter sowohl als einfaches CGI-BIN installieren, also diese bei quasi jedem Provider, der CGI-BIN unterstützt selber in das CGI-Verzeichnis kopieren und starten kann, und auch als APACHE Modul ausführen kann. Als Modul wird es vom Apache Server ein vielfaches schneller ausgeführt (ca. Faktor 20-50), da es in den Apache-Server fest eingebunden ist, also nicht bei jedem Zugriff extra gestartet werden muß. PHP3 Module werden abenfalls für alle gängigen Datenbanken angeboten. Wer mit weiteren Modulen für Apache arbeiten möchte, der kann sich auf der Site http://modules.apache.org über weitere Module informieren. Zur Dokumentation der Sprache PHP3 sei die Site http://www.php3.de empfohlen. Hier findet man alles vom Tutorial bis hin zu Befehlssammlungen zur Ansteuerung von **MySQL** unter PHP3. Auch Mailing-Listen, auf denen Probleme diskutiert werden, können über ein grafisches Benutzer-Interface abonniert werden.







37.2 Werkzeuge zum Programmieren

Um PHP zu programmieren und zu testen, braucht man nur eine Arbeitsstation mit Windows 95/98/NT und einen Internetzugang. Leider sind Editoren, wie Frontpage für PHP3 völlig ungeeignet, weil diese den Quellcode zerstören. Für unser kleines Tutorial genügt aber auch das WORDPAD. Das überspielen der Seiten kann entweder mit Netscape oder mit einem beliebigen FTP-Client geschehen. Falls Sie bisher Ihre Seiten nur mit Frontpage aktualisiert haben, dann benutzen Sie bitte das FTP-Interface, welches bei fast allen Providern ebenfalls genutzt werden kann.







37.3 Provider für MySQL und PHP

Um Provider in Ihrem Gebiet zu finden, geben Sie in die Suchmaschine http://www.metacrawer.com einfach die Begriffe **provider mysql php/fi München** oder einen anderen Stadtnamen an. Überregionale Provider bieten oft MySQL und PHP3 2.0/3.0 ohne Support an. Wer hier professionellen Support braucht, der sollte sich einen lokalen Provider suchen. Andernfalls bieten die Sites http://www.rent-a-database.de und http://www.puretec.de MySQL und PHP3 zur Nutzung an.







37.4 Installation der notwendigen Komponenten

Unser Beipiel bezieht sich auf LINUX. Es ist aber auch auf beliebige andere UNIX Server übertragbar. Der Installation unter Windows 95/98/NT ist ebenfalls möglich, jedoch läuft nach 30 Tagen die Nutzungsberechtigung aus. Auch liegen mir bisher noch keine genauen Informationen über die Stabilität der Datenbank unter Microsoft vor.






38. Installation von MySQL

Die folgenden Schritte können Sie sich evtl. ersparen, wenn Ihre LINUX Distribution MySQL vorinstalliert hat. **MySQL** ist bei fast allen Distributionen enthalten, und evtl. schon komplett vorinstalliert.

Als erster Schritt ist es notwendig, die aktuelle Version des MySQL-Servers aus dem Internet zu laden. Dazu gehen Sie auf die Homepage http://www.mysql.com Kopieren Sie die heruntergeladene Datei **mysql-VERSIONSNUMMER.tar.gz** (VERSIONSNUMMER steht für die jeweilige Versionsnummer der Software) in das Verzeichnis **/usr/local** und wechseln mit **cd** dorthin.

Geben Sie:

```
tar xzf mySQL-VERSIONSNUMMER.tar.gz
```

ein, um das Archiv zu dekomprimieren.

Falls es Schwierigkeiten geben sollte, dann schauen Sie sich bitte in das Kapitel Installationsanweisungen zu Beginn diese Handbuches an.

Im Verzeichnis **/usr/local** wurde ein Verzeichnis namens **mysql-VERSIONSNUMMER** angelegt. Wechseln Sie dorthin und geben Sie ein:

```
./configure
```

Dieses startet das GNU Software Paket zur automatischen Konfiguration des Quellcodes von **MySQL**. Da es viele verschiedene UNIX-Systeme gibt, die alle etwas andere Funktionen mit unterschiedlichen Parametern in den Systembibliotheken besitzen, ist also eine solches Programm eine wirkliche Erleichterung. Unter SOLARIS müssen Sie erst die entsprechenden Programme von der Site:

http://www.sunfreeware.com

laden und installieren. Eine detaillierte Anleitung liegt dort bei.

Geben Sie nun **./configure** ein. Es wird das sogenannte **Makefile** erzeugt, welches die Steuerdatei für GNU make, ist, dem eigentlichen Programm, welches den Compiler mit seinen vielen Optionen startet.

Geben Sie nun **make** ein und machen Sie sich eine Tasse Kaffee oder Tee.

In dem **Makefile** sind nun einige Standardverzeichnisse voreingestellt, die Sie aber mit dem Editior **joe** (Hilfen mit STRG-k h) nach Belieben ändern können.

Für erfahrene LINUX'ler sei noch erwähnt, daß der SQL-Server selbstverständlich auch im User Modus und auf anderen Ports installiert werden kann. Hierzu beachten Sie bitte die Erläuterungen im Quellcode. Nach diesen Änderungen kann man auf einem Server quasi beliebig viele eigenständige **MySQL** Server starten. Somit kann jede Domain auf dem Server ihren eigenen SQL-Server haben und administrieren. Geben Sie nun **make** ein. Die Binaries des MySQL-Servers werden kompiliert. Die Installation der Binaries wird durch **make install** durchgeführt. Bevor nun **MySQL** gestartet werden kann, muß die Benutzerdatenbank erstellt werden.

Geben Sie hierzu **./scriptsmysql_install_db** ein. Damit man später auch auf die Datenbank zugreifen kann, müssen die Zugriffsrechte für den User **wwwrun** gegeben werden. Hierzu loggt man sich als **root** ein, und gibt folgende Befehle ein:

```
mysql > insert into user
(host,user,password,select_priv,insert_priv,update_priv,delete_priv,create_priv,drop_priv,reload_priv,shutdown_priv,process_priv,file_priv)
values ('localhost', 'wwwrun', password(''), 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
'Y', 'Y', 'Y', 'Y');

mysql > insert into db (host,db,user,Select_priv,Insert_priv,Update_priv,Delete_priv,
Create_priv,Drop_priv)values ('localhost', '%', 'wwwrun',
'Y','Y','Y','Y','Y','Y','Y');
```

Nun die Syntax mag nicht jedem verständlich sein. Hierzu gibt es im Kapitel Privilegien weitere Informationen.

Danach kann mit dem Befehl: mysql.server start der Server gestartet werden.

Der Befehl **mysql.server stop** beendet den SQL-Server.

Soweit nun zu der Installation des **MySQL** Servers.








38.1 Installation von PHP 3.0 unter Windows 98

Die Installation von PHP 3.0 unter Windows 98 ist sehr einfach. Alle Beispiele funktionieren auch mit Windows ohne Einschränkungen. Allerdings muß Ihre Arbeitsstation mit einer Netzwerkkarte ausgestattet sein. Das schöne daran ist, daß Sie auf Ihrere Windows Arbeitsstation die Beispiele ausgiebig testen können, ohne einen Server aufbauen zu müssen. Das betrifft auch die Beispiele für Datenbanken und MySQL. Da der Apache 1.3.6 WWW-Server gestartet ist, können alle User im Netz Ihre auf Ihrer Arbeitsstation entwickelten Skripte testen und Verbesserungsvorschläge machen. Ist alles fertig, so kopieren Sie die Beispiele einfach auf Ihren WWW-Server - fertig. Wenn Sie nicht wissen, wo - http://www.rent-a-database.de/support/.

Zuerst benötigen Sie aus dem Internet das Binary für WIN-32 von der Site http://www.apache.org. Als zweites laden Sie sich das PHP 3.0.x Binary für WIN-32 von http://www.php3.net.

Für einfache Datenbanken und um die folgenden Beispiele testen zu können, benötigen Sie nicht mehr. Später können Sie sich noch MySQL für Windows laden und installieren. In den PHP 3 DLL's ist alles enthalten, was man z.B. auch zum Anschluß von SQL-Datenbanken oder für kleinere Datenbanken braucht, z.B. ist die Berkley Datenbankmaschine vollständig enthalten.

Der Apacheserver läßt sich einfach installieren, da er ein Installationsscript besitzt. Wählen Sie die "custom" Installation, installieren Sie keine Sourcecodes und als Installationsverzeichnis wählen Sie **c:\apache**. Die Installation läuft völlig von alleine ab.

Danach müssen Sie die PHP 3 Bibliotheken und das Binary installieren. Hierzu müssen sie mit WINZIP (Siehe www.tucows.com) die Dateien in ein Verzeichnis extrahieren. Legen Sie hierzu das Verzeichnis **c:\php3** an und extrahieren Sie alle Dateien aus dem PHP 3-Archiv hierhin.

Nun müssen noch einige Anpassungen vorgenommen werden. Die erste Anpassung betrifft die Datei **c:\windows\hosts.sam**. Benennen Sie diese in **hosts** um, und tragen Sie folgende Zeilen noch ein:

```
127.0.0.1       localhost
127.0.0.1       win98
10.0.0.1        win98
```

Anstelle von 10.0.0.1 müssen Sie die TCP/IP Nummer Ihrer Arbeitsstation eintragen. Falls Sie diese nicht kennen, so tippen Sie in der DOS-SHELL einfach: **ipconfig** ein. Die Ausgabe sagt Ihnen, welche IP-Nummer Ihr Host im Moment hat. In einigen Netzwerken mit NT-Servern werden IP-Nummern aus einem Pool vergeben. Das nennt man DHCP und ist völliger Mist, da Sie nun immer eine andere IP-Nummer nach jedem Neustart der Arbeitsstation erhalten. Wenden Sie sich hierzu an Ihren Systemadministrator und bitten ihn, für Ihre Arbeitsstation eine feste IP-Nummer im DHCP Server zu reservieren. Das ist ca. 1 Minute Arbeit (Wenn man sich auskennt). Alternativ vergeben Sie für Ihre Netzwerkkarte eine zusätzliche IP-Nummer, die dann fest eingetragen ist. Diese sollten Sie unbedingt aber mit Ihrem Systemadministrator vorher absprechen, da eventuell ansonsten ganze Server außer Gefecht gesetzt sind, oder Router und Firewalls verrückt spielen.

Im Verzeichnis c:\php3 befindet sich eine Datei **php3.ini-dist**. Diese müssen Sie in **php3.ini** umbenennen, und in das Verzeichnis **c:\windows** kopieren. Danach kopieren Sie alle Dateien aus dem Verzeichnis c:\php3 auch noch in das Verzeichnis **c:\apache\cgi-bin**. Das hat den einfachen Grund, daß Sie später keine Probleme mit der Portierung von PHP 3 Scripten auf einen Internet-Server bekommen, unbedingt notwendig ist es aber nicht.

Damit nun der Apache-Server genau bescheid weiß, wann er nun PHP3 Scipte vor der Auslieferung zu interpretieren hat, muß man ihn darüber informieren, daß z.B. Seiten mit der Endung **.php3** vor der Auslieferung an einen Browser noch durch den PHP 3 Interpreter geschickt werden müssen. Hierzu muß die zentrale Konfigurationsdatei des Apache, die es seit der Version 1.3.6 gibt, noch etwas ergänzt werden:

```
# AddType allows you to tweak mime.types without actually editing it, or to
# make certain files to be certain types.
#
# For example, the PHP3 module (not part of the Apache distribution)
# will typically use:
#
ScriptAlias     /php3/  "c:/php3/"
AddType application/x-httpd-php3 .php3
Action application/x-httpd-php3 "/php3/php.exe"

#AddType application/x-httpd-php3-source .phps
```

Dies sagt dem Apache-Server, daß ab sofort alle Dateien, die mit **.php3** enden, interpretiert werden müssen, und zwar muß dann die Datei **php.exe** automatisch gestartet werden. Im Gegensatz zu UNIX (Achtung, UNIX-Kenner !) ist hier die Zeile **Action .....php.exe..** zusätzlich enthalten, damit der Interpreter auch gestartet wird.

Danach müssen Sie die Arbeitsstation neu starten - dank Microsoft hat man des öfteren eine Kaffeepause.....

Wenn die Arbeitsstation dann neu gestartet ist, starten Sie mit [Start]->[Programme]->[Apache Web Server]->[Start apche as console app] den Apache WWW-Server mit PHP 3 Extensions.

Legen Sie im Verzeichnis **c:\apache\htdocs** nun eine Datei mit dem Namen: **php3info.php3** mit dem Inhalt:

```
<? phpinfo(); ?>
```

Nun öffnen Sie Ihren Browser und geben Sie als URL folgendes ein: http://win98/phpinfo.php3

Leider hat Windows 98 ein kleines Problem - Die Netzwerkroutinen sind alle von Einsteigern programmiert worden, und dementsprechend fehlerhaft. Das hat zur Folge, daß die Datei **/etc/hosts** nur nach einem mysteriösen Timeout auch abgefragt wird. Wenn Sie in Ihren Browser folgendes eingeben, dann kommt die PHP-Info-Seite wie aus der Pistole geschossen:

http://10.0.0.1/phpinfo.php3

Geben Sie hier Ihre eigene IP-Nummer an, und Sie sehen, alles funktioniert prächtig.

Von anderen Netzwerkstationen dürfen Sie natürlich diese WWW-Seite mit den PHP-Info's ebenfalls aufrufen. Ihre Arbeitstation ist nun der zentrale WWW-Server mit PHP 3 Extensions und Datenbank. Datenbank ? Am Ende des folgenden Tutorials finden Sie eine einfache Datenbank für alles Mögliche, welche ausschließlich in PHP 3 mit der in PHP 3 enthaltenen Berkley Database Engine programmiert wurde. Zusatzsoftware, Lizenzen - braucht keiner - alles da !

Nun viel Spaß mit dem kurzen Tutorial im übernächsten Kapitel !





38.2 Installation von PHP3 3.0.x unter LINUX

Einige Apache Versionen haben noch die etwas ältere Version von PHP3 (2.0) eingebunden, daher sollten Sie möglichst updaten. Die Scripte werden wohl ohne Probleme lauffähig sein, jedoch hat sich die Version 3.0 erheblich weiterentwickelt.

Da PHP3 3.0.x stets bei der neuesten APACHE Version als Modul enthalten ist, sollten Sie den aktuellen Apache-Server entfernen, um den neuen installieren zu können.

Hierzu laden Sie von der Site http://www.apache.org den neuesten Server und installieren diesen so, wie Sie auch MySQL installiert haben. Ein Problem besteht jedoch noch: Falls Sie möchten, daß der Apache-Server in einem von **/usr/local/httpd** abreichenden Verzeichnis installiert werden soll, dann Tippen Sie folgendes ein:

```
./configure -prefix=/usr/local/httpd
```

Damit wird der Apache-Server später im Verzeichnis **/usr/local/httpd** installiert. Nachdem dieses Skript beendet ist, geben Sie **cd ../php-3.0** ein. Starten Sie das Skript zum Konfigurieren von PHP mit:

```
./configure -with-mysql -with-apache=../apache_1.3.0 -enable-track-vars
```

Nachdem nun alles (hoffentlich korrekt) konfiguriert ist, geben Sie zum Erstellen von PHP3 einfach nur **make** ein und direkt anschließend, zum Installieren des Programmes, **make install** ein. Nun wechseln Sie wieder ins Apache-Verzeichnis (cd ../apache_1.3.0) und starten erneut das Konfigurationsskript mit dem Befehl:

```
./configure -prefix=/usr/local/httpd -activate-module=scr/modules/php3/libphp3.a
```

Zur Erklärung der vorangegangenen Befehle:

Bis hierher haben wir nun also zuerst die Install Skripte von Apache so konfiguriert, daß PHP3 3.0 in den Quellcode des Apache eingebunden wird. Danach wurde die PHP Bibliothek konfiguriert. Das mag vielleicht dem Microsoft infizierten Administrator etwas seltsam erscheinen, einem C-Programmierer ist diese vorgehensweise jedoch vertraut. Wem dies zu schwierig ist, der sollte sich die RPM-Dateien von S.u.S.E. oder RedHat LINUX aus dem Internet laden, und diese installieren. Die Vorgehensweise ist im Kapitel Installation von Binaries genau beschrieben.

Mit **make** kann nun der Apache-Server kompiliert werden. Wie oben bereits geschrieben, wird mit **make install** der Apache WWW-Server komplett im Verzeichnis **/usr/local/apache_1.3.x** installiert. Für SOLARIS User ist dieses Verzeichnis **/usr/local/bin**. Weitere Informationen zu Solaris finden Sie im Kapitel solaris. PHP3 3.0 verlangt nach einer Konfigurationsdatei, die stets im Verzeichnis **/usr/local/lib/php3.ini** abgelegt sein muß. Diese muß nur in besonderen Fällen verändert werden, im Allgemeinen ist also keine Änderung erforderlich.

Im Verzeichnis **/usr/local/httpd/conf/** liegen alle Konfigurationsdateien des Apache (httpd.conf). Seit der Version 1.3.x sind alle Konfigurationsdateien in der Datei httpd.conf zusammengefasst. Unter vielen LINUX Distributionen existiert ein Link im Verzeichnis **/etc/httpd/httpd.conf** auf diese Konfigurationsdatei.

Damit der Apache-Server vor der Auslieferung der Seite die PHP-Codes erkennen und interpretieren kann, muß diesem gesagt werden, daß alle Dateien auf dem WWW-Server, die die Endung **.php3** haben, zuerst interpretiert werden müssen. Falls man dieses nicht tut, erscheint plötzlich php3 Code auf der WWW-Seite. Um dieses Mißgeschick zu beheben, fügt man die Zeile:

```
AddType application/x-httpd-php3 .php3
```

hinzu. Wer möchte, daß auch Endungen mit **.phtml** oder z.B. mit **.asp** (zur Irreführung) interpretiert werden, der muß diese Endungen in derselben Zeile noch nach **.php3** anfügen.

Danach muß der Apache-Server neu gestartet werden. Hierzu gibt es inzwischen in allen LINUX-Versionen die Dateien **/etc/rc.d/apache** (nach dem LINUX FSSTD) die einfach mit **/etc/rc.d/apache start/stop/restart** den Apache-Server starten, stoppen oder neu starten.




mySQL




39. PHP3 3.0 Tutorial Schritt für Schritt

PHP3 ist eine Skriptsprache, die direkt in HTML-Seiten eingebettet wird, d.h. der Autor schreibt PHP3-Befehle zusammen mit HTML-Befehlen, JAVASCRIPT, u.s.w. in eine Datei und legt diese ganz gewöhnlich als HTML - Seite auf seinem WWW-Server ab.

Wird diese Datei von einem Betrachter mit dem Browser angefordert, so liest der WWW-Server, z.B. der APACHE Server die HTML - Seite von der Festplatte, erkennt anhand der Endung, daß darin ein Programm enthalten sein muß, und übergibt diese Seite zuerst einmal an den PHP3 Interpreter. Entweder dieser ist in den Apache Server einkompiliert, oder er exisitiert als eigenes Binary, welches dann zuerst gestartet werden muß. Da dieses Binary ca. 1.2 MByte groß ist, kann es also bei viel Servertraffic dazu kommen, daß die CPU und die Festplatte überlastet werden. Hierzu haben die Entwickler eine andere Lösung gefunden. Sie haben PHP3 als Modul in den Apache WWW-Server einkompiliert, sodaß nicht bei jedem Abruf einer Seite dieser neu gestartet werden muß. Der einkompilierte PHP3 Interpreter arbeitet die anfallenden Jobs dann aus einer QUEUE heraus ab. Somit kann in einigen Fällen der Apache Server um Faktor 5-100 schneller sein, als z.B. der IIS 4.0+ oder OmniWeb. Darum sollten Sie auch unter Windows NT stets den Apache WWW-Server verwenden. Für PERL wurde derselbe Weg beschritten, hier heißt das Modul FASTCGI. Alle großen Websites, wie Yahoo, Lycos, Altavista, HOTBOT (jaja!) arbeiten mit diesen Modulen und dem Apache WWW-Server.

PHP3 ist com Prinzip her ähnlich LIVEWIRE von Netscape/AOL oder ASP von Microsoft. Es gibt sogar einen ASP2PHP3 Konverter, den Sie in unserem Archiv auf http://www.rent-a-database.de/konverter/ finden. Es gibt auch ein ASP Modul für den APACHE WWW-Server, welches jedoch erhebliche Lizenzgebühren kostet. Somit stehen Ihnen alle Welten offen.

Fast alle Provider unterstützen PHP3. Leider gibt es aber erhebliche Unterschiede bei der Unterstützung der Libraries. Für PHP gibt es verschiedenste Module, die Routinen zur Ansteuerung von MySQL oder anderen Datenbanken enthalten. Von Interesse ist auch, daß Sie mit Hilfe von PHP Zugriff auf Sockets haben, um z.B. eine WHOIS Abfrage einer Domain über ein PHP Skript automatisieren können. Sie finden dieses Beispiel in unserem PHP3 Archiv. Falls also einige Beispiele nicht funktionieren, jedoch PHP prinzipiell funktionsfähig ist, dann sind nich alle Fähigkeiten von PHP unterstützt.

Je nach Installation interpretiert diese PHP-Zusatzsoftware nur Dateien mit der Endung ".php3", oder aber PHP3 3.0/4.0 Code in Dateien mit der Endung **.html, .htm.**. Hierzu beachten Sie bitte die Zusatzinformationen in dem Kopf der HTML-Seiten. Falls Sie sich hier nicht auskennen, so führ ein Studium der HTML Anleitung von Stephan Münz weiter.

PHP3 wird seit etwa 1994 entwickelt und erfreut sich stetig wachsender Beliebtheit. Es gibt Routinen für fast alle Datenbanken, die den direkten Zugriff von PHP3 auf den Datenbankserver erlauben. Die Sprache ist an BASIC, Java und Perl und C angelehnt.

Sie ist für BASIC Kenner sehr leicht lesbar, und man kann schnell Erfolge erzielen. Die

Objektorientierten Spracherweiterungen sind natürlich schwieriger zu erlernen. Die wichtigsten Sprachelemente, die Sie benötigen, um fertige Module zu nutzen, werden natürlich ausführlich erklärt. Eine ausführliche Beschreibung liegt bei www.php3.net. Das Handbuch ist ebenfalls unter der o.g. URL oder auf einem der deutschen Mirrors verfügbar: http://www.php3.net, oder http://de.php.net.

Nun viel Spaß mit PHP3 !








39.1 Tutorial PHP 3

PHP 3.0/4.0 ist eine Sprache, die sich grundlegend von **HTML** unterscheidet. Damit der Server diese vor der Auslieferung an den Browser interpretieren kann, muß er diese in dem HTML-Code erkennen können. Zur Makierung von PHP Scripten gibt es folgende Möglichkeiten, die vom Apache-Server erkannt werden:

<? ... ?>

<?php ... ?>

<script language="php"> ... </script>

<% ... %>







39.2 Grundsätzliches zum Tutorial

Das Tutorial setzt obige Installationen voraus. Alle PHP 3 Skripte werden mit der Endung **.php3** in dem Verzeichnis **c:\apache\htdocs** unter Windows 98 oder im normalen Verzeichnis Ihres WWW-Servers im Internet abgelegt. Alle Scripte sind auf beiden Systemen lauffähig, es könnte jedoch Unterschiede bei den in PHP 3.0 aktivierten Bibliothenken geben. Die Datei phpinfo.php3 gibt Ihnen alle Informationen, um herauszufinden, ob die Datenbankbibliotheken für z.B. mSQL, MySQL, Oracle oder der Berkley Datenbank eventuell nicht korrekt eingebunden sind. Auf unserer Site: **http://www.rent-a-database.de** haben wir natürlich dafür gesorgt, daß Sie einfach alle Skripte, die Sie auf Ihrer Arbeitsstation entwickelt und getestet haben, in das Internet kopieren können, und daß diese dann auch funktionieren. Bei anderen Anbietern muß dies nicht so sein. Man sollte auch wissen, daß die Endung **.php3** eventuell auch **.phtml** oder noch anderes lauten kann. Lesen Sie hierzu im Handbuch Ihres Providers nach ;-)







39.3 Ein erster Test von PHP 3.0/4.0

Nehmen Sie hierzu eine einfache, korrekte HTML-Seite, und betten Sie folgenden Code ein:

```
<?
        echo "Hallo";
?>
```

Merke: PHP - Befehle werden mit ";" abgeschlossen.







39.4 Sonderzeichen

Es gibt eine Reihe von Sonderzeichen. Wichtigste Sonderzeichen: "\n" und "\t": Neue Zeile und Tabulator

```
echo "Hallo\n";
echo "Hallo\n";
```

Es gibt viele weitere Sonderzeichen, wie z.B. \t, \r ... Schauen Sie bitte hierzu auf http://www.php3.net nach.







39.5 Variablen

Um sinnvolle Anwendungen erstellen zu können, benötigen wir Variablen. **Merke: Alle Variablennamen beginnen mit $**.

```
$text = "Ich bin ein String !";
echo $text;
echo ($text,$text,$text);
$l = "langer";
$k = "kurzer";
echo "Ich bin ein $l$l$l$l$l$l Text!";
echo "Ich bin ein $k Text!";
$i = 10;
$j = 5;
echo ($i,"+",$j,"=",$i+$j);
```

Der Typ der Variablen (ganze Zahl, Gleitpunktzahl, String) wird je nach Verwendung von PHP automatisch bestimmt. Schließlich ist PHP3 ja auch ein Interpreter. Bei Kompilern müssen die Datentypen vorher bestimmt werden.







39.6 Vordefinierte Variablen mit spezielle Funktionen

Bei der Übernahme von Eingaben aus Formularen an den PHP3 3.0/4.0 Interpreter oder an Datenbanken muß irgendwo abgefragt werden können. Schauen wir uns ein Beispiel an. Nehmen Sie hierzu eine korrekte HTML-Seite und betten Sie folgenden Code ein:

<FORM ACTION="ausgabe.html" METHOD=POST>

<INPUT NAME="eingabe">

<INPUT TYPE="submit">

</FORM>

Beim Klicken auf den Submit-Button wird also die Datei **ausgabe.html** aufgerufen. Diese HTML-Datei könnte dann folgenden Code enthalten:

```
<?
        echo "Sie haben $eingabe eingegeben. ";
?>
```

Daran kann man erkennen, daß PHP3 3.0/4.0 die Eingabe aus dem Formular in diejenige Variable zur Weiterverarbeitung ablegt, die im Formular im Feld **NAME** angegeben wurde. Unter Verwendung von weiteren Namen im Formular können natürlich auch viele Felder gleichzeitig oder nacheinander abgefragt werden.







39.7 Einfache Rechenoperationen

Folgende Rechenoperationen stehen in PHP3 zur Verfügung:

- "+": Addition, $i+$j
- "-": Subtraktion, $i-$j
- "*": Multiplikation, $i*$j
- "/": Division, $i/$j
- "%": Reste-Bildung, $i%$j: z.B. 23%17 ergibt 6, da 23 geteilt durch 17 gleich 1 Rest 6 ist.
- ".": Verknüpft Strings:

```
$l = "langer";
$k = "kurzer";
echo $l.$k;
```

ergibt: langerkurzer

Dazu kommen noch ein paar Abkürzungen:

- $i++ erhöht $i um 1.
- ++$i erhöht $i ebenfalls um 1.
- $i-- erniedrigt $i um 1.
- --$i erniedrigt $i ebenfalls um 1.

Der Unterschied zwischen $i++ und ++$i ist:

```
$i=0;
echo $i++;
```

gibt 0 aus, anschließend wird $i auf den Wert 1 erhöht.

```
$i=0;
echo ++$i;
```

erhöht zuerst $i auf 1 und gibt den Wert 1 aus.







39.8 Schleifen

Gleich ein Beispiel:

```
$t = "Ich bin eine while - Schleife<BR>\n";
$i = 0;
while ($i<100) {
        echo $t;
        $i++;
}
```

Hier wird 100 mal der Text in der Variablen $t ausgegeben. Zu Beginn wird $i auf 0 gesetzt. $i wird in jedem Schleifendurchlauf um 1 erhöht, bis $i den Wert 100 erreicht. Wenn die Bedingung ($i<100) nicht mehr wahr ist bricht die Schleife ab.







39.9 Weitere Befehle zur Ablaufkontrolle

Neben der obigen Schreibweise für **WHILE**:

```
$t = "Ich bin eine WHILE Schleife ! <BR>\n";
$i = 0;
while ($i<100) {
        echo $t;
        $i++;
}
```

kann man alternativ auch schreiben:

```
$t = "Ich bin eine DO-WHILE Schleife!<BR>\n";
$i = 0;
do {
        echo $t;
        $i++;
} while ($i<100);
```

Was ist der Unterschied ?

Die Schleife weiter unten wird mindestens einmal durchlaufen, da die Abfrage erst am Ende erfolgt, ein kleiner und wichtiger Unterschied.

Eine weitere Möglichkeit, eine Schleife zu programmieren, ist die von BASIC und C her altbekannte **for** Schleife:

```
$t = "Ich bin eine FOR Schleife !<BR>\n";
for ($i=0;$i<100;$i++) {
        echo $t;
}
```

Der **for** Befehl besteht aus drei Ausdrücken.

```
for (ausdruck1;ausdruck2;ausdruck3) { ... }
```

Mit ausdruck1 wird die Schleife initialisiert, d.h. normalerweise wird die Variable, die die Schleifendurchläufe zählt, auf den Anfangswert gesetzt.

ausdruck2 gibt die Abbruchbedingung an.

In ausdruck3 wird die Variable, die die Schleifendurchläufe zählt, erhöht oder erniedrigt.

Der **for** Befehl hat den Vorteil, daß alle zur Kontrolle der Schleife nötigen Befehle in einer Zeile

stehen. Hier ein weiteres Beispiel, diesmal wird aber heruntergezählt:

```
$t = "Ich bin eine FOR DOWNTO Schleife!<BR>\n";
for ($i=100;$i>0;$i--) {
        echo $t;
}
```

Wie Sie sehen, ist die Syntax stark an C angelehnt, ohne daß der Quellcode seine Lesbarkeit verliert. Aber auch BASIC Programmierer sollten sich schnell an die veränderte Systax gewöhnen können.



39.10 Fallunterscheidungen

Zur Fallunterscheidung gibt es den obligatorischen **if**-Befehl:

```
if ($i<0) {
        echo "$i ist kleiner als Null\n";
}
```

oder auch

```
if ($i<0) {
        echo "$i ist kleiner als Null\n";
} else {
        echo "$i ist nicht kleiner als Null\n";
}
```

Es ist bei allen Programmierarbeiten immer sinnvoll, daß Sie das **ELSE** immer mit hineinschreiben. Das erhöht zum Einen die Lesbarkeit für andere und und ermöglicht es Ihnen, Fehler schneller auffinden zu können, falls einmal eine Variable vom Wert her aus dem Ruder läuft .....

Man kann diesen Befehl auch schachteln:

```
if ($i<0) {
        echo "$i ist kleiner als Null\n";
} else if ($i>0) {
        echo "$i ist groesser als Null\n";
} else {
        echo "$i ist Null\n";
}
```

Hat man mehrere Tests der gleichen Variable, so kann man mit dem **switch** Befehl evtl. Arbeit einsparen:

```
switch ($name) {
        case "hallo":
                echo "Hallo Tutorial !";
                break;
        case "tag":
                echo "Guten Tag !";
                break;
        case "abend":
                echo "Guten Abend !";
                break;
```

```
        default:
                echo "Oder wir sind am Nordpol !";
}
```

Falls die Variable $name den Wert "hallo" hat, wird als nächster Befehl:

```
echo "Hallo Tutorial !";
```

ausgeführt. Normalerweise werden alle nachfolgenden Befehle auch ausgeführt, z.B. auch :

```
echo "Oder wir sind am Nordpol ! ";
```

Da dies meist nicht erwünscht ist, springt man mit **break** aus dem switch-Befehl heraus. Gewöhnen Sie sich diese Strukturen so an, daß Sie automatisch in Fleisch und Blut übergehen. Sie werden sich ansonsten wundern, wie schnell Sie Ihren eigenen Code nicht mehr lesen können.....








39.11 Vergleiche

Bisher konnten wir nur testen, ob $i<100 oder $i>100 gilt. Es gibt aber noch mehr Möglichkeiten:

```
$i==10: Ist $i gleich 10?
$i!=10: Ist $i ungleich 10?
$i>=10: Ist $i größer oder gleich 10?
$i<=10: Ist $i größer oder gleich 10?
```

Man kann auch kombinieren:

```
($i==10) && ($j>0): Ist $i gleich 10 und $j größer als 0?
($i==10) || ($j==0): Ist $i gleich 10 oder $j gleich 0?
```

Natürlich können Sie auch Strings so vergleichen, um diese z.B. sortieren zu können. Da PHP3 ein Interpreter ist, funktioniert natürlich auch das. Wir wollen aber hier nicht alle Eventualitäten durchspielen, sondern die Grundlagen erlernen und eventuell kleine Unterschiede zu bisherigen Sprachen herausstellen. Dem erfahrenen C-Programmierer wird dieses vielleicht alles völlig banal vorkommen, trozdem gibt es bei PHP3 3.0/4.0 kleine Unterschiede, die man sich als C-Programmierer merken sollte.







39.12 Wie verwende ich Funktionen und Bibliotheken ?

Eines der größten Vorteile von PHP ist das riesige Archiv von Unterprogrammen, Funktionen und Routinen, die fast alles abdecken, was man braucht oder gebrauchen könnte. Die Kunst ist es, das Rad hierbei nicht selber zu erfinden, sondern auf bewährte Programmteile von anderen Programmierern zurückzugreifen.

Operativer, blinder Aktivismus kann einen bei der Entwicklung im Vergleich um Tage zurückwerfen.

Auf dem Server http://www.php3.net findet man Bibliotheken zur Einbindung von Datenbanken, Cursor-Bibliotheken u.s.w. Bisher ist nicht weiter aufgefallen, daß z.B. der Befehl **echo** auch nur eine Funktion aus einer Bibliothek ist, nämlich der internen. Dieser gehört zu den elementaren Funktionen. Was uns jedoch mehr interessieren sollte, sind die zusammengesetzten Funktionen.

Im nächsten Abschnitt wird z.B. die Funktion **isset($name)** verwendet. Davor steht eine **if** Abfrage. Daraus kann man entnehmen, daß die Funktion **isset()** als Ergebnis entweder **true** oder **false** an die **if** Abfrage zurückliefert. Der Parameter, der ihr dabei übergeben wird, ist eine Stringvariable. Ist der String leer, so wird **false** zurückgegeben, falls der String gesetzt ist, wird **true** zurückgegeben. Allgemein kann man also die Funktion **isset** so beschreiben:

```
int isset(string)
```

Der Ausdruck int bezeichnet eine Zahl, oder **true** oder **false**. Intern werden die Boolschen Operatoren (true, false) als 1 oder 0 dargestellt, daher ist **int** korrekt.

Sie sollten sich an diese Schreibweise schnell gewöhnen, da fast alle Dokumentationen zu Routinen so verfaßt sind. Wer die Schreibweise nicht versteht, der kann auch nicht die Routinen fremder Programmierer in seine eigenen Programme einbauen.

Schauen wir uns kurz eine selbstgeschriebene Funktion an:

```
function funktion ($arg_1, $arg_2, ..., $arg_n) {
    echo "Beispielfunktion.\n";
    return $rueckgabewert;
}
```

Diese Funktion benötigt Übergabewerte als String und gibt einen String zurück, ist also ähnlich der Funktion isset, nur der Rückgabewert ist diesmal ein String.

In den folgenden Unterkapiteln werden Funktionen verwendet, die Datenbanken öffen, Daten auslesen, und die Datenbank schließen. Schauen wir uns also einmal auf http://www.php3.net/manual/ref.mysql.php3, dem Handbuch von PHP3 um. Hier steht:

```
int mysql_connect(string [hostname [:port] [:/path/to/socket] ] , string
[username] , string [password] );
```

Die Ähnlichkeit mit der obigen Funktion isset ist verblüffend. Man kann also z.B. auch schreiben:

```
<?php
```

```
$id = mysql_connect("localhost", "nobody", "");

if(!$id) {
    echo "Fehler beim öffnen der Datenbank\n";
    exit;
}
```

Überlegen Sie sich, wie Sie z.B. die Funktion **mysql_connect()** für andere Programmierer dokumentieren würden. Vergleichen Sie einfach mal Ihre Idee mit denen des der Programmierer der Routine im PHP3 Handbuch im Internet.

Wir haben nun Funktionen kennengelernt, die recht einfach zu verstehen sind.

Im Folgenden werden nun Klassen vorgestellt, die einfach nutzbare Routinen zur Verfügung stellen, sofern man einmal die Syntax verstanden hat. Diese Klassen erinnern ein wenig an C++ Containerklassen, sind aber sehr einfach zu nutzen, da einige in C++ überhaupt nicht verständliche Regeln auch nicht in PHP 3 übernommen wurden, z.B. Vererbungen, mehrfache Vererbungen, u.s.w.

PHP3 Klassen sind viel einfacher zu nutzen. Hier ein Beispiel:

```
class datenbank_funktionen {
    function such_was () {
        echo "Suche in Datenbank.";
    }
}

$neue_funktion = new datenbank_funktionen;
$neue_funktion -> such_was ();
```

Autsch ! Eine ungewohnte schreibweise ? Die Klasse **datenbank_funktionen** beinhaltet eine oder mehrere nützliche Unterroutinen (hier nur eine) einer Klasse **datenbank_funktionen**. Die Unterroutine heißt hier: **such_was**.

Es gibt tausende Programmierer, die unter PHP 3 tolle Routinen, ganze Datenbankinterfaces für SQL-Datenbanken geschrieben haben, und diese nun als Klasse der Allgemeinheit kostenlos zur Verfügung stellen.

Mit diesen Klassen läßt sich in mit nur einer einzigen Zeile eine SQL-Datenbank, z.B. **MySQL** ansprechen (wie oben gesehen). Die Kunst ist es, die Dokumentationen richtig zu lesen, und die Klassen richtig in PHP 3 anzusprechen. Alles andere geht dann wie von Geisterhand.

In den Zeilen nach **class** wurde eine Funktion definiert. Danach folgte ein Aufruf **$neue_funktion = new .....**

Dieses nennt sich Instanz. Damit aktiviert man quasi diese Funktion, damit der PHP 3 Interpreter etwas Speicher anfordern und die Funktionen von der Festplatte lädt.

Danach folgt eine recht ungewöhnliche Syntax (zumindest für BASIC Programmierer), die völlig Zeigern auf Funktionen der Programmiersprache **C** ähnelt. Genau das ist es auch in PHP 3. Ein Zeiger auf eine Funktion in einer Klasse. Innerhalb der Klasse können Funktionen auch andere Funktionen aufrufen. Hierzu muß die Klasse auf eine Funktion von sich selber zeigen. Dies kann man mit dem Zeiger auf die eigene Klasse erreichen. Dann würde die Funktion so heißen: **this->such_was();**. **this** ist also die Klasse selber.

Wenn man das nun behalten und verstanden hat, dann kann mit der Einbindung anderer Klassen nicht mehr viel schieflaufen, zumal der Rest der Programmiersprache PHP 3 doch sehr einfachem BASIC ähnelt. Aus diesem Grunde existiert auch ein Konverter von Microsoft Visual Basic, genauer gesagt den Active Server Pages (ASP) nach PHP 3.0 .

Wunderhübsch, da man nun neben dem kostenlosen Zugriff auf PHP 3.0 Archive auch noch alle ASP-Scripte einfach nach PHP konvertieren kann. Wie gesagt, PHP ist nicht ohne Absicht so konzipiert worden und bereits bei 1.2 Millionen WWW-Servern im Internet im Einsatz.

Den Konverter findet man übrigens hier: http://home.i1.net/~naken/asp2php/. Er läuft ausschließlich nur auf billigen, absturzsicheren Betriebssystemen, z.B. LINUX mit GTK Oberfläche.

Ebenfalls hier findet man einen Export-Filter der ACCESS Tabellenstruktur nach MySQL. Mit diesem Script lassen sich zur Beschleunigung von ACCESS durch MySQL die Tabellenstukturen einfach exportieren.

So nun für diejenigen, die es noch nicht erwarten können, in Klassenbibliotheken schon einmal hineinzuschnuppern, hier ein Tip: http://phplib.shonline.de/index.php3. Es werden aber noch viele Beispiele folgen, die auch ohne diese Klassen auskommen. Wer aber höchst ansprüchsvolle Internet-Sites entwickeln möchte, der muß sich diese Adresse unbedingt merken....






39.13 Vordefinierte Variablen

Da ja PHP3 Code und HTML stets nebeneinander in einer Datei zu finden sind, stellt sich natürlich die Frage nach einem Mechanismus, mit dem man Variablen aus Eingabeformularen an PHP3 übergeben kann.

Angenommen, die Datei **formular.html** enthält folgendes Formular:

```
<FORM ACTION="formular.html" METHOD=POST>
<INPUT NAME="name">
<INPUT TYPE="submit">
```

Dann ruft sich beim Klicken auf **submit** die HTML Datei mit dem PHP3 Code selbst auf. Allerdings ist dann die Variable **name** name gesetzt und kann ausgelesen werden. Dies können wir folgendermaßen ausnutzen:

```
<?
        if (isset($name)) {
                echo "Sie haben $name eingegeben\n";
        }

<P>

<FORM ACTION="formular.html" METHOD=POST>
<INPUT NAME="name">
<INPUT TYPE="submit">
```

Wird die Datei zum ersten Mal aufgerufen, d.h. ohne daß man auf den **submit** Button klickt, so ist die Variable $name nicht gesetzt, d.h. isset($beliebigername) ist falsch. Füllt man das Formular dann aus und klickt auf submit, so ist die Variable gesetzt, d.h. isset($name) ist wahr, und es wird "Sie haben $name eingegeben\n"; ausgegeben. Will man danach abbrechen, so sollte:

```
 exit;
```

verwendet werden. Binden Sie diese kleinen CODE Snipplets in eine HTML Seite ein, kopieren Sie diese auf den WWW-Server und sie sehen, daß es funktioniert.

PHP3 ist wie ein riesiger Modulbaukasten, nun fehlen uns nur noch die Bibliotheken zum Zugriff auf

Dateien und MySQL, damit wir schon einmal die Inhalte unserer MySQL Datenbank im Internet veröffentlichen können.....








39.14 Dateien einbinden

Der Befehl:

```
include("dateiname");
```

liest den Inhalt der Datei **dateiname** so, als ob er an dieser Stelle stehen würde. Damit kann z.B. einheitliches Layout bei einer größeren Anzahl von Dateien erreicht werden. HEADER und FOOTER lassen sich so ganz einfach realisieren. Sie können auch HTML Code hineinschreiben und das Layout Ihrer WWW-Seiten so einfach durch Änderung des HEADERS ändern. Das funktioniert deswegen, weil PHP3 Code immer vor dem HTML Code und auch JAVASCRIPT Code ausgeführt wird. Das ist sehr praktisch, da z.B. auch der Little-Idiot SHOP eine Inhalte aus einer MySQL Datenbank bezieht, und diese in JAVASCRIPT Code einbaut. So können sie einen Offline SHOP generieren, der aus einer SQL Datenbank die Artikel bezieht. Schauen Sie einmal unter http://www.little-idiot.de nach. (Klicken Sie einmal auf das Männchen ...)




39.15 Dateien lesen und schreiben

Für die Ausgabe der Suchergebnisse von **MySQL** z.B., ist es nützlich, diese in eine Datei zu schreiben, um dann in Ruhe duch die Ausgabe vor-und zurück blättern zu können. Das entlastet den MySQL Serverdämon. Moderne JDBC-Treiber unterstützen das Blättern mit CURSOR, hierzu muß jedoch die SQL-Datenbank die Cursor - Positionierung unterstützen. **MySQL** tut dies nicht, daher ist diese Möglichkeit ein **Workaround** für **Cursorpositionierung**. Weiterhin kann man hiermit auch alle Arten von Gästebücher .... programmieren.

Wichtig: Aus Sicherheitsgründen ist der Webserver einem Benutzer auf dem Server-Computer zugeordnet, der sehr wenig Rechte hat. U.a. darf er nicht einfach in eine Datei in ihrem Verzeichnis schreiben. Genausowenig darf er eine Datei in Ihrem Verzeichnis erzeugen.

OK. Gehen wir davon aus, daß wir nun eine Datei mit dem Namen xxx haben, deren Inhalt wir lesen wollen.

Zuerst müssen wir die Datei öffnen, dabei muß angegeben werden, was mit der Datei geschehen soll: Wollen wir lesen, schreiben, anhängen,... Der Befehl lautet:

```
$datei = fopen("xxx","r");
```

Das heißt, wir öffnen die Datei mit dem Namen xxx und wollen darin lesen ("r"). Die technischen Details lassen wir dabei von der Variablen $datei regeln, d.h. wenn wir später aus der Datei lesen wollen, so benützen wir die Variable $datei.

Nun lesen wir zeilenweise den Inhalt der Datei, und zwar so lange, bis wir ans Ende der Datei gelangt sind:

```
while (!feof($datei)) {
        $zeile = fgets($datei,1000);
        echo $zeile;
}
```

feof($datei) ist wahr, sobald wir an das Datei-Ende gelangt sind. $zeile = fgets($datei,1000); liest maximal die nächsten 1000 Zeichen, hört aber auf, sobald eine neue Zeile beginnt, oder das Ende der Datei erreicht ist.

echo $zeile; gibt einfach das Gelesene wieder aus.

Schließlich muß die Datei noch geschlossen werden:

```
fclose($datei);
```

Es gibt unterschiedliche Arten, eine Datei zu öffnen:

- "r": nur lesen, begonnen wird am Dateianfang.
- "r+": lesen und schreiben, begonnen wird am Dateianfang.
- "w": nur schreiben. Existiert die Datei bereits, wird der bisherige Inhalt gelöscht. Existiert sie nicht, wird versucht sie zu erzeugen.
- "w+": lesen und schreiben. Ansonsten wie "w".
- "a": nur schreiben. Begonnen wird am Ende der Datei (a wie append, anhängen). Existiert sie nicht, wird versucht, sie anzulegen.
- "a+": lesen und schreiben. Ansonsten wie "a". Achtung: Es wird beim Lesen natürlich auch am Ende der Datei begonnen (dort steht natürlich nichts mehr...)

Mit dem Befehl:

```
fwrite($datei, "Ein Text");
```

kann der String Ein Text in die Datei geschrieben werden.






39.16 Bildbearbeitung

Mit PHP3 können wir nicht nur HTML-Text erzeugen, wir können sogar Bilder dynamisch aus der Datenbank liefern. Zum Beispiel können wir eine Datei "bild.php3" erstellen, die ein Bild zurückliefern wird. In diese Datei schreiben wir keine HTML-Befehle, sondern ausschließlich PHP-Befehle. Ein Beispiel:

```
<?
  Header( "Content-type: image/gif");
  $im = imagecreate(200,100);
  $blue = ImageColorAllocate($im, 0x2c,0x6D,0xAF);
  ImageRectangle($im,0,0,200,100,$blue);
  ImageGif($im);
  ImageDestroy($im);
?>
```

Ruft man diese Datei auf, so erscheint ein blaues Rechteck. Tut es das nicht, dann hat Ihr Provider vergessen, die Grafikroutinen in den PHP Interpreter einzubinden. Dann können Sie natürlich so kleine Dinge, wie dynamisch beschriftetet Buttons nicht realisieren. Auch eine Bilddatenbank, die automatisch Bilder in andere Pixelformate konvertiert (und abrechnet), lassen sich dann so nicht realisieren.

Natürlich kann diese Datei auch mit:

```
<IMG SRC="bild.php3">
```

in eine HTML-Seite eingebunden werden. Was bewirken obige Befehle ?

Der Befehl:

```
Header( "Content-type: image/gif");
```

teilt dem Browser mit, daß nun ein GIF-Bild kommen wird. (Um Fehlermeldungen in unserer PHP3-Datei angezeigt zu bekommen, können wir diesen Befehl auch zeitweise auskommentieren, indem man davor ein #-Zeichen setzt.)

Der Befehl:

```
$im = imagecreate(200,100);
```

erzeugt ein Bild, das 200 Pixel breit und 100 Pixel hoch ist. Jede Farbe, die wir verwenden wollen, muß zuerst erzeugt werden. Dies geschieht mit dem Befehl:

```
$blue = ImageColorAllocate($im, 0x2c,0x6D,0xAF);
```

Dabei bezeichnet $im das Bild, für das wir die Farbe erzeugen wollen, während die restlichen drei Parameter die hexadezimalen RGB-Werte sind. (Man beachte die Schreibweise: 0x2c ergibt den dezimalen Rot-Anteil 36.)

Mit dem Befehl:

```
ImageRectangle($im,0,0,200,100,$blue);
```

zeichnen wir uns ein Rechteck, das bei 0,0 (links oben) beginnt und 200 Pixel nach rechts und 100 Pixel nach unten endet, und in der eben erzeugten Farbe $blue gefüllt ist.

Der Befehl:

```
ImageGif($im);
```

erzeugen zuerst ein GIF-Bild aus der internen Darstellung und sendet es an den Betrachter, während

```
ImageDestroy($im);
```

den internen Speicherplatz für das Bild wieder freigibt.

Es gibt nun eine Reihe von weiteren Möglichkeiten. Zum Beispiel kann noch ein Text in das Bild geschrieben werden:

```
$white = ImageColorAllocate($im, 255,255,255);
imagestring($im, 5, 20, 20, "Hallo",  $white);
```

Der erste Befehl erzeugt weiße Farbe: Man sieht, die RGB-Werte können auch als Dezimal-Werte angegeben werden. Der zweite Befehl schreibt den Text "Hallo" in Schriftgröße 5 beginnend bei Position 20,20 (20 Pixel nach rechts, 20 Pixel nach unten). Die beiden Befehle sollten direkt vor ImageGif($im); stehen.

Wie Sie sehen, können Sie nun dynamisch Bilder generieren, oder Schrift in Bilder bei der Auslieferung hineinkopieren. So könnten Sie jedes Bild für jeden einzelnen Kunden quasi handsignieren !

Vielleicht sagt Ihnen **GIMP** noch nichts. GIMP ist ein ADOBE PHOTOSHOP Clone, der unter UNIX und Windows läuft. GIMP ist mit denselben Routinen geschrieben worden, die auch für PHP3

verfügabr sind. Sie können also auch die Routinen für Weichzeichner, BLUR ...u.s.w. in PHP3 ansprechen.






39.17 Zugriff auf Datenbanken

Nun sind wir über kleine Umwege endlich am Punkt Datenbanken angelangt .....

Um eine SQL-Abfrage mit PHP 3 auszuführen, muß zuerst die Datenbank geöffnet werden (vergleichbar mit dem Zugriff auf eine Datei), dann wird die SQL-Befehlszeile an die Datenbank geschickt, die Antwort der Datenbank wird aufgenommen und schließlich wird die Datenbank-Verbindung wieder geschlossen.

Zuerst müssen wir uns etwas Hintergrundwissen über die SQL-Befehlszeile verschaffen:

SQL Befehle zur Erinnerung

SQL-Datenbanken bestehen aus 1 oder mehreren sogenannten Tabellen. Jeder Datensatz der Datenbank ist genau eine Zeile in einer Tabelle. Beispiel Ein Stadtführer für Köln:

```
Die Tabelle Stadt:

ID      Name            Art             Note            Kommentar
1       POPOKATEPETL    Biergarten      1               Sehr schöner Baumbestand
2       CHLODWIG-ECK    Biergarten      1               Gute Bratwürste
3       ...             ...             ...             ...
```

Man kann nun auf eine dieser Tabellen zugreifen, aber auch Tabellen verküpfen. Wir wollen uns hier der Einfachkeit halber auf eine einzige Tabelle beschränken.

Es gibt im wesentlichen vier Abfragemöglichkeiten:

1. Auslesen: SELECT
2. Einfügen: INSERT
3. Überschreiben: UPDATE
4. Löschen: DELETE

Will man z.B. alle Namen, die sich in der Datenbank befinden, auslesen, so gibt man den SQL-Befehl:

```
SELECT Name FROM Stadt;
```

ein. Will man Name und Art der Kneipe ausgeben, so lautet der SQL-Befehl:

```
SELECT Name,Art FROM Stadt;
```

Soll schließlich die ganze Spalte Stadt der Tabelle ausgelesen werden, so genügt der Befehl:

```
SELECT * FROM Stadt;
```

Die auszulesenden Datensätze können nun noch weiter spezifiziert werden:

```
SELECT * FROM Stadt WHERE NOTE=1;
```

gibt nur die Datensätze aus, die in der Spalte Note eine 1 enthalten. Sortiert werden können die Datensätze mit:

```
SELECT * FROM Stadt WHERE NOTE=1 SORT BY Name;
```

werden. Neue Datensätze werden folgendermaßen hinzugefügt:

```
INSERT Stadt (Name,Art,Note,Kommentar) VALUES
('Glenk','Biergarten','1','Gute Bratwürste');
```

Vorhandene Datensätze können mit:

```
UPDATE Stadt SET Note='2',Kommentar='Die Bratwürste lassen nach' WHERE
ID=2;
```

überschrieben werden.






39.18 SQL-Befehle mit PHP 3

Die obigen SQL-Befehle verwenden wir nun mit PHP 3. Wir wollen als Beispiel alle Datensätz aus der Tabelle kneipen in der Datenbank www2 ausgeben.

Das öffnen einer Datenbankverbindung ist ebenso trivial, wie eine Datei zu öffnen. Achten Sie vielleicht einmal auf die Parallelen !

Zuerst muß eine Verbindung von PHP 3 zum Datenbank-Server (d.h. dem Computer, auf dem die Datenbank liegt) hergestellt werden:

```
$verbindung = @mysql_connect("localhost","nobody","");
```

localhost bedeutet, wir verbinden uns zum gleichen Computer, auf dem sich auch der Webserver befindet. Für den Fall, daß Sie sich über Internet an eine bestehende Datenbank anknüpfen möchten, so geben Sie den gültigen Hostnamen oder die IP-Nummer ein.

nobody ist ein gültiger Benutzername für diese Datenbank (dies wurde vom Datenbank-Verwalter vielleicht so vergeben...).

Die "" bedeuten, daß dieser Benutzer kein Passwort benötigt. Wir können überprüfen, ob die Verbindung funktioniert:

```
if (!$verbindung) {
        echo "Keine Verbindung möglich!\n";
        exit;
}
```

Wenn Sie nun diese Code Snipplets korrekt in eine HTML Seite mit PHP Code Zeichen ! einfügen, so eröffnen Sie über PHP eine Datenbankverbindung. Mit dem Befehl **netstat** auf dem Datenbankserver oder der Maschine, auf der PHP3 läuft, können Sie die Einzelheiten der Verbindung (Portnummer...) genau sehen.

Nun müssen wir die Datenbank nur noch auslesen ....







39.19 Weitere SQL-Befehle

Nachdem wir zuvor ja ein wenig Erfahrung mit SQL-Befehlen sammeln konnten, trauen wir uns auch an den DELETE-Befehl heran. Wichtig ist für den diesen Befehl, daß ganz genau (und vorsichtig) spezifiziert wird, welcher Datensatz oder welche Datensätze aus der Datenbank gelöscht werden sollen. In unserem Beispiel ist es am besten, wenn man über das Feld ID den zu löschenden Datensatz angibt, denn dieses Feld enthält für jeden Datensatz eine eindeutige Nummer. Wollen wir den Datensatz mit ID=120 löschen, so geben wir als Abfrage an:

```
$abfrage = "DELETE FROM kneipen WHERE ID=120";
```

Achtung: Vergißt man den WHERE-Teil, werden alle Datensätze gelöscht! Als Ersteller einer dynamischen Webseite mit PHP und einer Datenbank sollte man also immer genau darüber nachdenken, was den Betrachtern erlaubt werden soll und was nicht.

Als Übung sollten die einfach einmal einige INSERTS ausprobieren. Sie können einfach alle Kommandos aus dem Einsteiger Tutorial MySQL unter LINUX ausprobieren, indem Sie Sie einfach in den HTML bzw. PHP Code Ihrer HTML Seite hineinkopieren, und diese dann mit dem Browser aufrufen. Sie werden sehen, daß alles funktioniert. Sie können Datenbanken anlegen, Userrechte vergeben, Daten aus Dateien einlesen, Daten ergänzen und löschen. Alles über das PHP3 Interface. Die SQL Statements ist identisch mit denen, die Sie z.B. über den MySQL Client an den Datenbankserver gesandt haben.







39.20 Funktionen

Ein wichtiger Bestandteil einer jeden Programmiersprache ist die Fähigkeit, mehrere Befehle zu einem einzigen Befehl zusammenzufassen, d.h. neue Funktionen zu definieren. PHP3 hat diese Fähigkeit selbstverständlich auch.

Was in **MySQL** im Prinzip fehlt, sind sogenannte **stored procedures**. Diese kann man jedoch mit PHP3 einfach ersetzen. Das folgende Beispiel zeigt den Einsatz von Funktionen unter PHP3. Hier können Sie einfach die Routinen zum Ansprechen der MySQL Datenbank einbauen, und fertig sind Ihre "Stored Procedures".

Wir wollen für eine Anzahl von Tagen berechnen, wieviele Sekunden diese Tage zusammen haben:

```
function tag ($anzahl) {
        $sekunden = 3600 * 24 * $anzahl;
        return $sekunden;
}

$x = tag(7);
echo "7 Tage bestehen aus $x Sekunden !";
```

$anzahl nennt man auch Argument der Funktion. Eine Funktion kann mehrere Argumente, oder auch kein Argument besitzen:

```
function zinseszins ($prozent,$laufzeit) {
        $kapital = 10000.0;
        $x = 1.0 + ($proz/100.0);
        $y = $kapital * exp( $laufzeit*log($prozent) );
        return $y;
}
$proz = 6;
$zeit = 10;
$x = $zinseszins($proz,$laufzeit);
echo "Das Kapital beträgt $x Euro";
```

Mehr zu den verwendeten mathematischen Funktionen kann man im Handbuch PHP3 nachlesen.
Problematisch ist im obigen (zugegebenermaßen etwas konstruierten) Beispiel, daß die Variable **$kapital** in der Funktion selbst festgelegt wird. Natürlich wird man auch diesen Wert normalerweise als Argument der Funktion übergeben, also etwa so:

```
function zinseszins ($prozent,$laufzeit,$kapital) { ...
```

Wir wollen aber nun den Fall betrachten, daß diese Variable bereits gesetzt ist, und wir sie nicht als Argument übergeben wollen. Dann muß angeben werden, daß die Variable "von außen" stammt. Dies geschieht durch den Befehl **global**. Den Wirkungsbereich von Variablen nennt man **SCOPE**....:

```
$kapital = 10000.0;
function zinseszins ($prozent,$laufzeit) {
        global $kapital;
        $x = 1.0 + ($proz/100.0);
        $y = $kapital * exp( $laufzeit*log($prozent) );
        return $y;
}
$proz = 6;
$zeit = 10;
$x = $zinseszins($proz,$laufzeit);
echo "Das Kapital beträgt $x DM/EURO/YEN/US$..";
```






39.21 String - Funktionen

Ein weiteres wichtiges Betätigungsfeld von PHP3 ist die Bearbeitung von Zeichenketten. Es gibt eine ganze Reihe von Befehlen um Zeichenketten zu manipulieren, wir stellen hier nur die wichtigsten vor, der Rest kann im Handbuch im Kapitel "String-Funktionen" nachgeschlagen werden. trim entfernt Leerzeichen, Tabulatorzeichen etc. vor dem ersten und nach dem letzen "echten" Zeichen in der Zeichenkette.

```
$x = "   Ich bin eine Zeichenkette! ";
$y = trim($x);
```

Die Befehle ltrim() und chop() entfernen whitespace vor, bzw. nach der Zeichenkette.

Für die verspielten PHP-Programmierer:

strtoupper() und **strtolower()** wandeln alle Zeichen eines Strings in Großbuchstaben, bzw. Kleinbuchstaben um. **Achtung:** Probleme kann es bei Umlauten geben.

Der Befehl ucfirst() wandelt das erste Zeichen eines Strings in Großbuchstaben um (wenn möglich).

Der Befehl ucwords() hingegen wandelt in jedem Wort eines Strings das erste Zeichen in Großbuchstaben um (wenn möglich).

Der folgende Befehl ist besonders wichtig, wenn man vorhandene Text für das WWW aufbereiten will:

```
$x = nl2br($y);
```

ersetzt alle Zeilenumbruch-Zeichen ("\n") des Strings $y in die Zeichenfolge
\n.

Daneben gibt es noch einige wichtige Funktionen, um Strings zu vergleichen, ersetzen und mit HTML Sonderzeichen umzugehen. Schauen Sie sich hierzu im Handbuch zu PHP3 im Internet um.




39.22 Cookies - Mechanismen

Als Cookies werden kurze Informationen bezeichnet, die über den Browser auf die Festplatte des Clients abgelegt werden. Se sind vor allem beim elektronischen Einkaufen im WWW von Bedeutung, genauer gesagt sind sie immer dann von Bedeutung, wenn der Server weitergehende Information vom Betrachter haben muß, als nur die Internet-Adresse, die eine Seite anfordert. Dem Sender von Cookies ist es so möglich, zu erfahren, welche Internet-Seiten Sie besucht haben, welche Waren im Warenkorb liegen, u.s.w. Eine Firewall verhindert nicht unbedingt, daß Cookies auf Ihrer Festplatte landen.

Cookies bestehen aus folgenden Informationen:

```
Name:           Name
Value:          Wert
expires:        Verfallsdatum
domain:         Adressraum der Server, die das Cookie wieder lesen dürfen.
path:           Nur über Seiten, die in diesem Verzeichnis, oder Unterverzeichnissen
davon liegen, darf das Cookie angefordert werden.
secure:         Wenn gesetzt, darf das Cookie nur über verschlüsselte
                Informationskanäle übertragen werden.
```

Beispiel: Wir wollen feststellen, ob ein Betrachter eine Seite schon einmal aufgerufen hat. Dazu setzen wir beim ersten Aufruf der Seite ein Cookie. Dies muß allerdings vor dem HTML-Kopf geschehen, d.h. am Anfang der Datei.

```
<?
        $t = time()+3600*24*10;
        SetCookie("mycookie","Schon besucht",$t,"/test",".de");
?>
<HTML>
...
```

Dieser Befehl setzt ein Cookie mit dem Namen mycookie auf den Wert "schon besucht" mit einem Verfallsdatum von 10 Tagen. Lesen darf dieses Cookie jeder Server mit der Endung .de, allerdings nur aus Dateien, die in einem Verzeichnis /test, oder Unterverzeichnissen davon liegen.

Das Verfallsdatum des Cookie muß in Sekunden seit dem 1. Januar 1970 angegeben werden. Die Funktion time() gibt die Sekunden seit dem 1. Januar 1970 der momentanen Zeit an, wie übrigens alle UNIXe von dem 1. Januar 1970 an hochzählen. Weitere Datums- und Zeitfunktionen sind im Handbuch im Kapitel Datum/Zeit-Funktionen nachzulesen. Da die Berechnung der Sekunden nicht immer ganz übersichtlich ist, empfiehlt sich die Verwendung einer Funktion:

```
        function tage ($anzahl) {
                $sekunden = time()+ 3600 * 24 * $anzahl;
                return $sekunden;
        }
        SetCookie("mycookie","Schon besucht",tage(10),"/test",".de");
```

Cookies lesen ist mit PHP 3 wieder sehr einfach realisiert: Der Inhalt des Cookies ist in einer Variablen mit dem Namen des Cookies enthalten:

```
        echo "$mycookie";
```

gibt also den Wert des Cookies aus, in unserem Fall den Text "Schon besucht".

Ein Server kann mit PHP 3 ein Cookie wieder entfernen, indem lediglich der Befehl setcookie mit dem Namen des Cookies als Argument angegeben wird:

```
        SetCookie("mycookie");
```

Variable Variable

Beim Auslesen von Arrays in Formularen gibt es einige Tricks, die man kennen muß. In manchen Fällen will man Schleifen über eine Menge von Variablen laufen lassen. Zum Beispiel wenn man viele Eingabefelder in einem Formular hat:

```
<INPUT NAME="feld0" >
<INPUT NAME="feld1" >
<INPUT NAME="feld2" >
...
<INPUT NAME="feld99" >
```

Nun ist es sehr mühsam, alle Felder einzeln zu behandeln. Will man z.B. einfach den Inhalt aller Eingabefelder ausgeben, so kann man statt:

```
echo "feld0: $feld0 <BR>\n";
echo "feld1: $feld1 <BR>\n";
...
echo "feld99: $feld99 <BR>\n";
```

auch folgendermaßen vorgehen. Wir basteln uns den jeweiligen Variablennamen mit Hilfe der o.a. Stringfunktionen:

```
for ($i=0;$i<100;$i++) {
        $v = "feld".$i;
        echo "$v: $$v \n";
}
```

Mit einem $v erhalten wir den Variablennamen, mit $$v greifen wir auf den Inhalt der Variablen mit dem Namen $v zu.








39.23 Arrays aus Formularen

Abschließend noch ein letzter Trick: Wenn wir in einem Formular aus einer Menge von HTML-Checkboxen mit PHP ermitteln wollen, welche Checkboxen angeklickt wurden, dann scheitert die bisherige Vorgehensweise:

```
<INPUT TYPE=CHECKBOX NAME="x" VALUE=1 > Option 1
<INPUT TYPE=CHECKBOX NAME="x" VALUE=2 > Option 2
...
```

Schickt man dieses Formular ab, so hat die PHP-Variable $x nur den Wert der ersten angeklickten Checkbox. Der Trick ist, in HTML die Checkbox zum Beispiel so zu benennen:

```
<INPUT TYPE=CHECKBOX NAME="x[]" VALUE=1 > Option 1
<INPUT TYPE=CHECKBOX NAME="x[]" VALUE=2 > Option 2
...
```

Dadurch wird in PHP ein Array mit dem Namen $x existieren. Die Werte können dann so ausgelesen werden:

```
for ($i=0;$i<sizeof($x);$i++) {
        echo "$x[$i] ";
}
```

Was das mit MySQL zu tun hat ? Sehr viel, schloeßlich möchten Sie ja die Inhalte der MySQL Datenbank als HTML Seite übergeben. Wenn nun neben jedem Datensatz ein Button steht: "Ändern", der mit einem Formularfeld verknüpft ist, dann können die so recht einfach ein PHP3 Skript bauen, mit welchem Sie ausgewählte Datensätze ändern können.

Natürlich können wir hier nicht lange Skripte einfügen. Schauen Sie sich bitte in unsererm PHP3 Skriptarchiv um. Ein schönes Beispiel, wie es funktinieren kann, ist das Administratiostoolkit **phpMyAdmin**, welches Sie ja im Quellcode genau studieren können.







39.24 Ein paar Beispiele zu MySQL und PHP3

Wir möchten hier trotzdem ein paar kleine, lauffähige Skripte zu MySQL zeigen. Diese sollen Ihnen als einfachste Grundlage dienen und Sie ermutigen, sich auch einmal an etwas aufwendigere Aufgaben heranzutrauen.

Das Thema ist: **Abfragemaske für Bestellungen und Datenbankzugriff via MySQL und Ausgabe in HTML-Tabelle; Rückkopplung mit der Abfragemaske:**

LISTING 1a

```
<![cdata[<? require "header.html" ?>]]>

<![cdata[<h1>Abfrage Ihrer Bestellungen]]>

<![cdata[<form action="ausgabe.php3" method="POST">

<p>Kundennummer:
<p>
<![cdata[<input type="text" name="kn" value="<? print $kn ?>">]]>

<p>Datensatznummer:<p>
<![cdata[<input type="text" name="dn">]]>
<p>
<![cdata[<input type="submit" value="Datenbank abfragen">]]>
<p>
<![cdata[]]>
<p>
<![cdata[<? require "footer.html" ?>]]>
<p>

LISTING 1b  <p>
<tscreen><verb>

<?
  require "header.html";

  $connect=@mysql_pconnect($host,$user,$passwd);

  $query="select * from table where kn='$kn' and dn='$dn'";

  $result=@mysql_db_query($database,$query,$connect);
?>
```

```
<table cellpadding=5 cellspacing=2 border=0>
<tr bgcolor="#66cc66"><th>Anzahl<th>Preis

<?
  while ($row=mysql_fetch_array($result)){
    printf("<tr bgcolor=\"#cc%scc\">",($i++%2?"ff":"cc"));
    printf("<td>%s<td>%s",$row["anzahl"],$row["preis"]);
  }
?>



<p>

Anderer <a href="eingabe.php3?kn=<? print rawurlencode($kn)
?>">Datensatz.

<? require "footer.html" ?>
```

Listing2a: Objekte: einfacher Warenkorb:

```
<?
  class warenkorb{
    var $artikel;

    function rein($bez,$anz) {
      $this->artikel[$bez]+=$anz;
    }

    function raus($bez,$anz) {
      if ($this->$artikel[$bez]>=$anz)
        $this->artikel[$bez]-=$anz;
    }

    function liste($pre,$post="<br>") {
      while (list($bez,$anz)=each($this->artikel))
        printf("%s%s: %d%s",$pre,$bez,$anz,$post);
    }
  }

  class mywarenkorb extends warenkorb {
    var $identifikation;

    function personalisieren($id) {
      $this->$identifikation=$id;
    }
```

```
    function mywarenkorb() {
      $this->rein("kostenloser Katalog",1);
    }
  }

  $einkauf=new mywarenkorb;

  $einkauf->personalisieren("Niels");

  $einkauf->rein("Sonnencreme",1);
  $einkauf->rein("Grillfleisch",10);
?>

<p>Einkauf fnr das Grillen am Werdersee:
<ul type="circle">

<?
  $einkauf->liste("<li>","");
?>
```

Dynamische Benennung: Aufruf von Funktionen:

LISTING 2b

```
<?
  function implizit() {
    print "<p>Implizit.";
  }

  function explizit() {
    print "<p>Explizit.";
  }

  function funktion($call="implizit") {
    $call();
  }

  funktion();

  funktion("explizit");
?>
```






39.25 Systemnahe Funktionen: Arbeiten mit Sockets

Dieses Beispiel erfragt den Besitzer einer Domain. Es könnte sein, daß der Port 43 eventuell durch die Firewall gesperrt ist, über Modem/ISDN oder auf Ihrem WWW-Server funktioniert das Beispiel tadellos.

```
<pre>
<?
  $socket=fsockopen("whois.ripe.net",43,&$n,&$s);

  fputs($socket,"little-idiot.de\n");

  while (!feof($socket))
    print fgets($socket,512);

  fclose($socket);
?>
```

Dieses Beispiel macht über ein PHP3 Skript das, was Sie auch auf der WWW-Seite des NIC http://www.nic.de in der WHOIS Datenbank manuell erfragen können:

Name, Anschrift, Telefonnummer, Faxnummer, E-Mail über alle Inhaber von Domains. Nun stellen Sie sich einmal vor, sie hätten alle Domainnamen in Ihrer MySQL Datenbank. Haben Sie nicht ? Nun, da können wir helfen: Geheimtip: kobold.firmen-info.de. Bitte nicht so stressen .... Alternativ können Sie auch unter UNIX folgendes eingeben, allerdings sollten Sie viel Zeit und Speicherplatz haben....

```
user01@tunix:~/SGML/mysql > nslookup
Default Server:  tunix
Address:  x.x.x.x

>server ns.nic.de
Default Server:  [x.x.x.x]
Address:  x.x.x.x

>domain=.de
...

>ls -d > allededomains
>
```

Ich möchte nicht zu genau werden, da vermutlich ansonsten der DENIC Server

zusammenbricht....Wer möchte, der kann alle Anschriften, Telefonnummern, Faxnummern .... monatlich aktuell auf CDROM von mir erhalten. Zusammen mit der Steuerdatei, die alle Domainnamen enthält kann man dann über ein einfaches PHP3 Skript (ich habe es in PERL hier, wer hat es in PHP3 ?) und einwenig Fleißarbeit des Servers kann man dann in Richtung Online Marketing steuern. Allerdings sollte man hiermit keine Massenfaxe oder Massenmailings starten, wenn man nicht ein ganz böses Echo der Internet-Gemeinde wegen Verstoß gegen die Nettiquette haben möchte. Sie können sicher sein, daß Sie einige Monate dann auf E-Mails und Ihren WWW-Server verzichten müssen....



39.26 Was für ein WWW-Server läuft auf dieser Domain ?

Hier ein kleines Beispiel für ein Skript, welches den Typ von WWW-Server ausfindig macht, der eine Domain beherrbergt, genauso wie auf WWW.NETCRAFT.COM

```
<html>
<body>
<?PHP
/*
 * Function WebServerSpy created to check, which kind of WebServer is
 * running (e.g. NS Fasttrack, Apache etc.).
 * (c) 1999 by: Bernhard Ostheimer (bernhard.ostheimer@uni-mainz.de)
 * View Example @
 * http://service.wiwi.uni-mainz.de/~ostheimer/php/webserverspy/
 */

function WebServerSpy($ServerURL)
{
  $filepointer = fsockopen($ServerURL,80,&$errno,&$errstr);
  if(!$filepointer)
  {
    $WebServer="Error: $errstr ($errno)<br>\n";
  }
  else
  {
    fputs($filepointer,"GET / HTTP/1.0\n\n");
    while(!feof($filepointer))
    {
      $WebServer=fgets($filepointer,4096);
      if (ereg("^Server:",$WebServer))
      {
        $WebServer=trim(ereg_replace("^Server:","",$WebServer));
        break;
      }
    }
    fclose($filepointer);
  }
  return($WebServer);
}
print "<b>www.little-idiot.de  "; print
WebServerSpy("http://www.little-idiot.de");
?>
```







39.27 Überprüfung von erlaubten Buchstaben mit EREG()

Cracker versuchen stets, an ein eventuelles PERL-Script noch ein **;system(rm -rf *)** zu übergeben, damit Teile der Festplatte gelöscht werden. Daher sollte jede Eingabe durch User im Formular unbedingt noch einmal durch die Funktion **ereg()** auf erlaubte Buchstaben untersucht werden. EREG() sollte man überall unbedingt einsetzen.

```
function validate_email ($address) {
  return (ereg('^[-!#$%&\'*+\\./0-9=?A-Z^_A-z{|}~]+'.
               '@'.
               '[-!#$%&\'*+\\/0-9=?A-Z^_A-z{|}~]+\.'.
               '[-!#$%&\'*+\\./0-9=?A-Z^_A-z{|}~]+$',
              $address));
}
```







39.28 Upload einer Datei

Wie einfach man eine Datei von der Fetplatte auf den WWW-Server laden kann, zeigt dieses Beispiel: Bei der maximalen Größe und dem Zielverzeichnis sollte man jedoch die INI-Datei von PHP 3.0 beachten, und ggf. anpassen. Unter WIN32 funktioniert das Script nicht, unter UNIX hingegen sehr gut.

```
<html>

<? if ($userfile && $userfile!="none") { ?>
<center>
<h1> Your File:
<?

        echo "File: $userfile<br>";
        echo "Remote File: $userfile_name<br>";
        echo "Size: $userfile_size<br>";
        echo "Type: $userfile_type<br>";

 }
?>

<FORM ENCTYPE="multipart/form-data" ACTION="upload.html">
<INPUT TYPE="hidden" name="MAX_FILE_SIZE" value="100000">
Send this file: <INPUT NAME="userfile" TYPE="file">
<INPUT TYPE="submit" VALUE="Send File">
```







40. Projekt Einfacher Datenbankzugriff mit MySQL und PHP

Dieses Projekt beschreibt einen einfachen Zugriff auf eine MySQL Datenbank mit PHP3.

Nach der Einführung in die Sprache PHP3 wird hier nun anhand eines einfachen Beispiels genau beschrieben, wie man mit PHP3 auf die MySQL Datenbank zugreift. Dieses Projekt könnte als Ausgangsbasis für Erweiterungen oder neue Projekte sein.







40.1 Schema des Datenbankzugriffs

Das Schema eines Datenbankzugriffes ist recht einfach:

1. Verbindung zum MySQL-Server herstellen
2. Datenbank auswählen
3. SQL-Query abschicken
4. Ergebnis des Query verarbeiten
5. Verbindung beenden







40.2 Der erste Kontakt

Bevor man überhaupt eine Anfrage bei einem SQL Server stellen kann, muß man diesen zuerst kontaktieren.

Um eine Verbindung aufzubauen, muß man in der Regel drei Sachen wissen: den Namen und Port des Rechners, auf dem der Datenbankserver läuft, eine Benutzerkennung und das dazugehörige Paßwort. War die Verbindung erfolgreich, bekommt man von PHP3 einen sogenannten **link identifier** zurück, ähnlich dem Filehandle, den man bei Operationen auf der Datenbank mit übergeben muß.

```
<?
$link = mysql_pconnect("host:port","username","password");
?>
```







40.3 Auswahl der Datenbank

Nun muß man eine Datenbank auswählen, da ein SQL Server viele Datenbanken gleichzeitig beherbergen kann. Das entsprechende Paßwort zum Server sollte man zuvor kennen.

SQL-Anfragen beziehen sich jedoch immer nur auf eine ausgewählte davon. Damit MySQL weiß, welche Datenbank gerade gefragt ist, muß zuerst eine Datenbank ausgewählt werden. Alle folgenden Queries beziehen sich dann darauf, und zwar solange, bis eine neue Datenbank ausgewählt wird. Diese Queries sind an das Handle geknüpft. Somit können andere User auf andere Datenbanken des MySQL Servers zugreifen.

```
<?
mysql_select_db("database",$link);
?>
```







40.4 Anfrage an MySQL

Nun kann man die Datenbankabfrage starten. Das Rückgabe des Ergebnisses erfolgt auf demselben Wege, auf welchem die Anfrage gestartet wurde.

Ist eine Verbindung hergestellt und eine Datenbank ausgewählt, kann eine Abfrage gesendet werden. Dazu formuliert man zuerst eine SQL-Abfrage und schickt diese dann ab. Als Ergebnis erhält man einen sogenannten **result identifier** zurück, über den dann im folgenden die Werte des Ergebnisses abgefragt werden können:

```
<?
$query = "select name, vorname from tabelle where id > 10";
$result = mysql_query($query,$link);
?>
```

Konnte die Abfrage nicht ausgeführt werden, enthält **$result** einen negativen Wert.

War die Abfrage erfolgreich, können nun über den **result identifier** die zurückgelieferten Werte ausgewertet werden. SELECT-Abfragen liefern Tabellen zurück, die nun in einer Schleife Zeile für Zeile abgerufen werden können.

Dazu sollte man zuerst feststellen, wie viele Zeilen das Ergebnis enthält und dann in einer Schleife jede Zeile einzeln abrufen. Am einfachsten ist es jedoch, die komplette Zeile als Array abzufragen. Der Index der Array-Felder entspricht den Spaltennamen in der SELECT-Abfrage:

```
<?
$num = mysql_numrows($result);
for ($i = 0; $i < $num; $i++) {
        $row = mysql_fetch_array($result);
        echo "Name : $row[name], Vorname: $row[vorname]<br>";
}
?>
```







40.5 Schließen der Verbindung

Schließlich wird die Verbindung zum Datenbankserver geschlossen. Nur bei sehr vielen Zugriffen pro Minuten könnte es sich lohnen, die Verbindung offen zu lassen. Wenn Sie einen Datenbankserver über das Netzwerk nutzen, sollte Sie sich darüber im klaren sein, daß MySQL auf ca. 100 gleichzeitige, offenen Verbindungen begrenzt ist. Wenn Sie mehr simultane Verbindungen wünschen, so müssen Sie MySQL neu kompilieren. Der Quellcode ist leider nur unter LINUX bzw. UNIX verfügbar.






40.6 Übersicht der verwendeten Funktionen

Hier nun die Übersicht über die verwendeten Funktionen:

- **mysql_pconnect(host,user,password)** Stellt eine Verbindung zum MySQL-Server her

  Parameter:

  host: IP-Adresse oder DNS-Name des Servers, evtl. mit Port

  user: MySQL-Username

  password: zum Usernamen passendes Paßwort

  Rückgabe: link identifier
- **mysql_select_db(database,link)** Wählt eine Datenbank aus, die über diesen link identifier angesprochen wird

  Parameter:

  database: Name der auszuwählenden Datenbank

  link: ein von mysql_pconnect() zurückgelieferter link identifier

  Rückgabe: keine
- **mysql_query(query,link)** sendet über einen vorher geöffneten link eine SQL-Abfrage an den Datenbankserver

  Parameter:

  query: String, der die SQL-Abfrage enthält

  link: ein von mysql_pconnect() zurückgelieferter link identifier

  Rückgabe: ein result identifier, über den auf das Ergebnis der Abfrage zugegriffen werden kann
- mysql_numrows(result) liefert die Anzahl von Zeilen, die von einer SELECT-Anfrage zurückgegeben wurden

  Parameter:

  result: der result identifier der Abfrage, die untersucht werden soll

  Rückgabe: die Anzahl der Zeilen
- **mysql_fetch_array(result)** liefert eine Zeile (einen Datensatz) aus einem Ergebnis einer SELECT-Abfrage und springt danach zur nächsten Zeile

  Parameter:

  result: der result identifier der Abfrage, die untersucht werden soll

Rückgabe: ein Array, das die Werte der Zeile enthält. Die Schlüssel des Arrays entsprechen dabei den im SELECT verwendeten Spaltennamen.






40.7 Ein größeres Beispiel

Hier noch einmal ein komplettes Beispiel mit allen Details zur Veranschaulichung der o.a. Funktionen:

```
<? require("kopf.phtml"); ?>

<h1>Liste der Artikel

<table>

<tr><td><b>Autor<td><b>Artikel

<?
   $link = mysql_pconnect("localhost","print","vsys");
   mysql_select_db("print",$link);

   $query = "select autor, subject from artikel order by autor asc";
   $result = mysql_query($query,$link);
   $num = mysql_numrows($result);

   for ($i = 0; $i < $num; $i++) {
      $row = mysql_fetch_array($result);
      echo "<tr><td>$row[autor]<td>$row[subject]";
   }
?>



<? require("fuss.phtml"); ?>
```







40.8 Tips und Tricks

Hier noch abschließend einige Tips und Tricks zur Übergabe von Werte an den MySQL Server.

Werden in SQL Strings angegeben, müssen sie immer in einfachen Quotes stehen, also z.B.

```
$wert1="hallo";
$wert2="huhu";
$query = "INSERT INTO test VALUES(null,'$wert1','$wert2')";
```

Werden Spalten angefordert wie UNIX_TIMESTAMP(a.datum), so müßen sie auch unter diesem Namen abgefragt werden. Werden Spalten angefordert, wie a.id und b.id, können sie nicht unter a.id bzw b.id angefordert werden, sondern nur unter id. Um beide Werte anforden zu können, muß der Umstand ausgenutzt werden, daß die Spalten von 0 bis n durchnummeriert werden. Eine Abfrage wie diese hier z.B.:

```
$query = "SELECT a.id,b.id,a.verfasser,UNIX_TIMESTAMP(a.datum),b.bezeichnung
FROM anzeigen a,kategorie b WHERE a.kategorie=b.id";

$result = mysql_query($query,$link);
```

schreibt beispielsweise für den ersten Record die folgenden Werte in das Array **$row**:

```
Key => Value
id  => 5                          Wert von b.id
0   => 1                          Wert von a.id
1   => 5                          Wert von b.id

verfasser => harald

2   => harald

UNIX_TIMESTAMP(a.datum) => 925730559

3   => 925730559

bezeichnung => Software

4   => Software
```

Auf a.id kann also nur über $row[0] zugegriffen werden.







40.9 Fehlerkontrolle

Zum Abfangen von Fehlern sollte man stets auch die Rückgabewerte der Query und die Zahl der betroffenen Reihen (Tupel) ermitteln. Das erspart einem viel Suche im Falle dessen, das Suchergebnis falsch ist:

mysql_query(): Wenn ihr eine UPDATE- , INSERT- oder DELETE- Abfrage abschickt, ergibt diese Funktion eine 1 bei Erfolg, einen leeren String bei einem Misserfolg.

mysql_affected_rows($link) ergibt die Anzahl der betroffenen Records nach dem letzten **INSERT, UPDATE** oder **DELETE** Statement.




41. Projekt Adressen - Datenbank

Dieser kurze Workshop zeigt, wie einfach es ist, ein WWW-Interface mit Anschluß an **MySQL** zu progrmmieren. Das Ergebnis ist eines von tausenden Adressbüchern, die man überall finden kann. Für ein unternehmensweites Adressbuch mit ACCESS, Outlook und Winword-Anbindung fehlt dann nur noch auf jedem Client der ODBC-Treiber, hierzu jedoch später mehr.

Ein WWW-Interface hat gegenüber vielen anderen Frontends viele Vorteile:

1. Es ist von überall aus zu erreichen
2. Es kann Passwortgeschützt werden
3. Es können über Router und Firewalls der Zugang beschränkt werden
4. Es sind keine Lizenzen für Clients nötig
5. Es sind keine Serverlizenzen notwendig (PHP3 ist kostenlos)
6. Es ist zuverlässig
7. Es kann mit ODBC ...an andere Datenbanken angebunden werden (ACCESS)
8. Es kann mit JDBC Interfaces nachgerüstet werden, damit Cursorpositionierung möglich ist

Damit Sie keine Probleme bekommen, sollte Sie einige Dinge vorher überprüfen. Hier also die Checkliste:

- Es müssen der Apache-Server mit PHP3 3.0 und MySQL installiert und gestartet sein
- Sie müssen Zugang zu einem (Ihrem) Homeverzeichnis haben, in welchem normalerweise die HTML-Seiten abgelegt werden
- Der Apache-Server muß PHP3 3.0 interpretieren können. Das können Sie testen, wenn Sie ein einfaches Beispiel von der Site http://www.php3.net auf Ihren WWW-Server kopieren und diese mit dem Browser abfragen.
- Sie müssen eventuell Telnet-Zugang zu dem Server haben, um die MySQL Befehle absetzen, und die Datenbank starten zu können. Falls Ihr Provider dieses nicht anbietet, so installieren Sie ihr eigenes LINUX und die Software, oder schauen Sie sich auf http://www.rent-a-database.de um.

Nun also los !

Melden Sie sich zunächst unter Linux als User **root** an, und rufen Sie **mysql** auf. Mit folgender Befehlszeile können Sie die Adress-Datenbank **adressen** erstellen:

```
mysql > create database adressen;
```

Fertig ! Zum Verlassen geben Sie quit oder exit ein.

Beenden Sie MySQL und starten Sie die Datenbank, allerdings mit dem Parameter **adressen**.

Die Datenbank **adressen** ist nun standardmäßig Ihre aktuelle Datenbank, solange Sie nicht den Befehl aus dem Unterkapitel USE verwenden, oder MySQL nicht mehrere Datenbanken enthält.

Starten Sie nun MySQL erneut, allerdings mit dem Parameter **adressen**. Danach befinden Sie sich in der Datenbank.

Um nun eine Tabelle mit Spalten und Zeilen in der Datenbank anzulegen, geben, starten Sie erneut den **MySQL** Client mysql und geben Sie folgendes ein:

```
create table adressen (id INTEGERAUTO_INCREMENT PRIMARY KEY,
firma CHAR(40), name CHAR(25), vornameCHAR(25), anschrift CHAR(40),
telefonCHAR(40), fax CHAR(40));
```

Sie haben nun mehrere Spalten angelegt: Eine laufende Nummer "id", nach denen die Daten eindeutig zugeordnet werden "primary key", und die Spalten "firma", "name", "vorname", "anschrift", "telefon" und "fax". Wenn Sie möchten, können Sie auch gleich noch die Spalten "email" und "url" anlegen, was heutzutage ja schon üblich ist. Mit quit oder exit können Sie mysql wieder beenden.

Nun werden die SQL Statements in die HTML Seiten geschrieben. Hierzu legen Sie mit einem einfachen Editor eine Datei an:

```
<html>
<head>
<title>Eingabe

<body>
<script language="php">
<?

>
```

Der PHP3 Code muß mit <? beginnen, und mit einem > enden. Dazwischen werden die PHP3 Befehle geschrieben.

Schauen Sie sich die Dokumentation auf http://www.php.net/manual/phpfi2.html an. Hier finden Sie für alle gängigen Datenbanken die genaue Syntax der Funktionen, um Datenbanken anzusteuern, darunter auch die von **MySQL**. Ein Beispiel:

```
<?

mysql_connect("localhost",wwwrun);
$result = mysql("adressen","select * from adressen order by name");

>
```

SQL-Abfrage ein Wert zugewiesen, der es der Datenbank-Engine ermöglicht, auf die eigentlichen Datensätze zuzugreifen. Der erste Ausdruck in der Klammer definiert die Datenbank, aus der die Datensätze ausgelesen werden sollen. In unserem Fall ist dies die Datenbank **adressen**. Die zweite Anweisung enthält den eigentlichen SQL-Befehl. **select** bedeutet, daß Daten aus einer Tabelle gelesen werden sollen. ***** bedeutet, daß alle Datensätze ausgelesen werden sollen. **from adressen** bedeutet, daß die Daten aus einer Tabelle (in unserem Adreßbuch ist es die Tabelle **adressen**) ausgelesen werden sollen. Die Anweisung **order by name** ist nicht unbedingt notwendig, erhöht allerdings die Übersichtlichkeit der späteren Ausgabe. Er veranlaßt den Datenbank-Server, die Daten nach dem Wert des Feldes **name** zu sortieren.

Im nächsten Schritt gilt es zu ermitteln, wie viele Datensätze gefunden wurden:

```
 $num = mysql_numrows($result);
```

$num ist wiederum eine Variable. Sie enthält einen numerischen Wert (Integer), der die Anzahl der durch die **select** -Anweisung gewählten Datensätze enthält.

Jetzt wird, wenn gewünscht, die Anzahl der gewählten Datensätze ausgegeben.

echo "</P>";echo "$num Einträge gefunden!";echo "</P>"

Der Skript-Befehl **echo** bewirkt, daß der nachgestellte Ausdruck in der HTML-Datei ausgegeben wird. In der ersten Zeile ist das der HTML-Befehl, um einen Textabsatz zu erzeugen. In der zweiten Zeile wird zuerst die Variable "$num" ausgegeben, und der Text " Einträge gefunden! ". Zeile drei fügt am Enden wieder einen Absatz ein. Die Anzeige lautet also: **99 Einträge gefunden!**

Im nächsten Schritt definieren Sie eine neue Variable $i, die Sie später als Zähler benötigen. $i = 0; Jetzt programmieren Sie die eigentliche Abfrage:

```
while($i<$num){echo "<P>"; echo mysql_result($result,$i,"firma");echo "\n";echo
mysql_result($result,$i,"vorname");echo "\n";echo
mysql_result($result,$i,"name");echo "<BR>";echo
mysql_result($result,$i,"anschrift");echo "<BR>Telefon: ";echo
mysql_result($result,$i,"telefon");echo "<BR>Fax: ";echo
mysql_result($result,$i,"fax");echo "<BR>";$i++;};
```

Mit dem Ausdruck **while** erzeugen Sie eine Schleife, deren Inhalt sich zwischen { und } befindet. Die Angabe in der Klammer unmittelbar hinter dem While-Ausdruck gibt die Bedingung der Schleife an. Frei übersetzt heißt das: Mach, was zwischen den { }-Klammern steht, solange bis $i kleiner ist als $num. $num ist, wie schon erwähnt, die Variable, welche die Gesamtanzahl der gefundenen Datensätze enthält. $i wird einfach bei jedem Schleifendurchlauf um 1 erhöht "$i++". Die Schleife selbst sorgt für die Ausgabe der Daten in Tabellenform.

echo "<P>"; erzeugt einen Absatz, damit die Daten nicht direkt hintereinander ausgegeben werden.

echo mysql_result($result,$i,"firma");echo mysql_result($result,$i,"vorname");

Dieser Befehl gibt die Daten aus der Datenbank aus und setzt den jeweiligen Wert des Feldes »firma«, »vorname«, »name«, »anschrift«, »telefon«, »fax« ein. »mysql_result« legt fest, daß ein Wert aus der Datenbank ausgegeben wird. Die Daten in der Klammer geben an, daß der Wert des Feldes der SQL-Abfrage »$result« mit dem Index »$i« und dem Namen »firma« ausgegeben werden sollen.

Zuletzt beenden Sie das Skript mit </SCRIPT>

SQL-Datenbank unter Linux installieren. Die Eingabe-Seite Für die Eingabeseite sind zwei HTML-Seiten notwendig: Die erste erfaßt die Daten und gibt sie an die zweite Seite weiter. Diese speichert die Daten in der Datenbank. 13. Die beiden Dateien heißen in unserem Beispiel »input_form.htm« (Eingabeformular) und »input.php3«. Das Eingabeformular besteht nur aus einer einfachen Formulareingabe und einer Tabelle, die alle Felder des Formulars positioniert. Diese Zeilen werden, wie auch bei den PHP3-Skripten, einfach in den BODY-Tag der HTML-Datei eingebunden. Details zu HTML-Befehlen finden Sie auf der Homepage von SelfHTML.

```
<B>Bitte geben Sie die Daten für den neuen Adressdatensatz ein:
<FORM ACTION="input.php3" METHOD="POST">
<TABLE BORDER=0><br>
<TR><br>
        <TD>Firma:
        <TD><INPUT TYPE="text" NAME="firma" SIZE="40">


        <TR>
        <TD>Vorname:
        <TD><INPUT TYPE="text"
                NAME="vorname" SIZE="40">


        <TR>
        <TD>Name:

        <TD><INPUT TYPE="text" NAME="name">

<TR> <TD>Anschrift:
<TD><INPUT TYPE="text" NAME="anschrift">

<TR>      <TD>Telefon:
<TD><INPUT TYPE="text"
NAME="telefon">
<TR>     <TD>Fax:
<TD><INPUT
TYPE="text" NAME="fax">
<TR>      <TD>
<TD><INPUT
TYPE=submit VALUE="Abschicken">
<INPUT TYPE=reset VALUE="Löschen">
```

Nun fehlt noch die Datei zum Speichern der Daten in der Datenbank. Hier können Sie einen großen Teil der Abfrageseite wiederverwenden. Als erstes definieren Sie

```
 $result = mysql("adressen","insert into adressen (firma, vorname, name, anschrift,
telefon, fax) values ('$firma', '$vorname', '$name', '$anschrift',
'$telefon', '$fax')");   <p>
```

Dieser SQL-Befehl schreibt den neuen Datensatz in die Datenbank. Beenden Sie das Skript mit </SCRIPT>.

Die Such-Seite

Auch für die Funktion zum Suchen verschiedener Einträge, benötigen Sie wieder zwei HTML-Dateien. Eine zur Eingabe der Suchkriterien und eine zum Auswerten. Wir verwenden in diesem Beispiel die Dateinamen "suche.htm" zur Eingabe und "suche.php3" zur Ausgabe der Suchkriterien.

Die Eingabeseite

In dieser Datei binden Sie die folgenden Zeilen innerhalb des BODY - Tags in Ihrer Webseite ein. Damit erzeugen Sie ein Formular, in dem mit Hilfe einer Tabelle die verschiedenen Eingabefelder positioniert werden. Gesendet werden die eingegebenen Daten mit der POST-Methode an die Datei suche.php3:

```
<FORM ACTION="suche.php3" METHOD="POST">
<TABLE BORDER=0>
        <TR>      <TD>Suchbegriff :
        <TD>        <INPUT TYPE="text" NAME="suchbegriff" SIZE="40">
        <TR>      <TD>      <TD>        <INPUT TYPE=submit VALUE="Abschicken">
<INPUT TYPE=reset VALUE="Löschen">
```

Die Auswertungsseite basiert auf der Abfrageseite. Sie müssen nur die SQL-Abfrage folgendermaßen anpassen:

```
$result = mysql("adressen","select * from adressen where firma='%$suchbegriff%'or
vorname='%$suchbegriff%' or name='%$suchbegriff%'
or anschrift='%$suchbegriff%' or telefon='%$suchbegriff%'
or fax='%suchbegriff%' order by name");
```

Diese Befehlszeile verwendet eine einfache SQL-Select-Anweisung, die sich auf die Suchkriterien der Eingabeseite bezieht. Das %-Zeichen bewirkt, daß immer nur ein Teil der Zeichenkette eingegeben werden muß. Geben Sie beispielsweise als Suchbegriff »hallo« an, werden alle Datensätze angezeigt, die den Wortteil »hallo« enthalten.

Es gibt mehrere Möglichkeiten, Einträge in der Datenbank zu löschen: per Hand mittels des Programms mysql adressen, über den MySQL-ODBC-Treiber oder auch über eine Webseite. Unser Beispiel beschreibt das Löschen über eine Webseite.

Sollten Sie sich für die Variante via ODBC interessieren, lesen Sie im Kapitel ODBC nach. Zum Löschen von Datensätzen brauchen Sie zwei verschiedene HTML-Seiten. In der ersten Datei delete.php3 wird abgefragt, welche Datensätze zu löschen sind. Diese HTML-Datei übergibt die Werte dann mittels des POST-Befehls an die Datei »delete_action.php3«, die den Datensatz dann löscht und die Aktion bestätigt.

Die Auswahlseite zeigt den zu löschenden Datensatz an. Sie ähnelt einer normalen Abfrageseite. Die Datensätze werden aber nicht wie bei der Abfrageseite nur angezeigt, sondern als Hyperlink dargestellt. Bei der Aktivierung des Hyperlinks wird der gewählte Datensatz gelöscht.

```
while($i<$num){echo "<A HREF=delete_action.php3?id=";echo
mysql_result($result,$i,"id");echo ">";echo
mysql_result($result,$i,"firma");echo "<BR>";echo
mysql_result($result,$i,"vorname");echo " ";echo
mysql_result($result,$i,"nachname");echo "";echo "<BR>";echo
mysql_result($result,$i,"anschrift");echo "<BR>Telefon: ";echo
mysql_result($result,$i,"telefon");echo "<BR>Fax: ";echo
mysql_result($result,$i,"fax");echo "
";$i++;};
```

Die zweite Datei löscht den Datensatz. Die Verbindung mit dem Server übernehmen Sie aus dem Beispiel. Um den über die Auswahlseite (delete.php3) übergebenen Datensatz zu löschen, müssen Sie die folgende SQL-Anweisung einbauen:

$result = mysql("adressen","delete from adressen where id='$id'");

Dieser Aufruf löscht alle Datensätze aus der Datenbank **adressen** in der Tabelle »adressen«, bei denen der Wert »$id« dem von der Auswahlseite übergebenen Wert entspricht. Durch die Verwendung des Parameters »$id« wird nur ein einzelner Datensatz gelöscht, da Sie das Feld »id« als »auto_increment« definiert haben. Dadurch fügt sich dieser Wert bei jedem Datensatz automatisch ein, und kann nicht doppelt vorkommen. Verwenden Sie anstelle des Parameters »$id« zum Beispiel »$vorname«,

werden alle Einträge, bei denen der Vorname dem angegebenen Wert entspricht, gelöscht.

Das Ergebnis der Aktion geben Sie so aus: echo "Der Eintrag mit der Nummer $id wurde gelöscht!"; Diese Zeile meldet dem Anwender, daß der Eintrag mit der Nummer »$id« gelöscht wurde. Eine Abfrage nach positivem oder negativem Ergebnis beim Löschen des Datensatzes muß nicht mit eingebunden werden, da bei korrekter Funktionsweise des Servers keine Fehler auftreten können. Sollte allerdings der MySQL-Server einen Fehler feststellen, gibt er automatisch eine Fehlermeldung aus. Diese Funktion können Sie später noch beliebig ausbauen, indem Sie sie etwa mit einer Suchmaschine verbinden. So können Sie beispielsweise eingeben, daß alle Datensätze mit dem Vorname »Benjamin« gelöscht werden sollen.








42. Nützliche PHP 3.0/4.0 Bibliotheken

Boris Erdmann und Kristian Köhntopp, Kiel haben ein paar sehr nutzliche PHP 3.0 Bibliotheken (phplib_6) für folgende Anwendungen geschrieben:

- Shopping Cart mit PostgreSQL und MySQL
- Authentifizierung
- Session Tracking für Shops (Kennung für User)
- Session Management (Verwaltung von Zugriffsberechtigungen bei SHOP)
- Zugriffsberechtigungen zu bestimmten Seiten
- Dynamische Diagramme
- Verwaltung von Bäumen unter HTML

Diese Routinen glänzen durch hervorragende Qualität und sind auch recht ausgereift, wie die große Referenzliste auch zeigt. Das besondere an diesen Routinen ist, daß z.B. das Session Management für SHOP auch ohne Cookies läuft, da es die IP-Nummer des Surfers als Kennung verwendet.

Zu finden unter http://phplib.shonline.de/







43. Erstellung der ER-Diagramme

Die ER - Diagramme dienen dazu, daß der Datenbankplaner sich über die Abhängigkeiten der Datensätze gründlich klar wird. Mit den Normalformen von E.F. Codd hat dies direkt nichts zu tun. Einem ER - Diagramm kann man Zusammenhäge zwischen den Datensätzen und die Aufteilung der Daten in unabhängige Tabellen entnehmen.






43.1 Darstellung der Entitätstypen und Relationen

In den Diagrammen werden die Entitäten, die Beziehungen der Entitäten untereinander, also den **relationships** und die **Komplexität** der Beziehung aufgezeichnet. Der Übersicht halber hat man sich auf bestimmte Diagrammtypen geeinigt:

Entitätstypen werden durch ein Recheck dargestellt, und die zugehörigen Attribute elliptisch umrahmt. Zur Übung hier ein Beispiel:

```
                    Buch

          InvNr   Titel   Autor   Verlag  ISBN
```

Das Buch wird natürlich mit einem Rechteck umrahmt, die Attribute (der Rest) mit einer flachen Ellipse. Mehrwertige Attribute, z.B. der Autor werden mit einer doppelten, flachen Ellipse umrahmt. Der Grund liegt einfach darin, daß viele Bücher einen Co-Autor haben. Bei der Erfassung in der Datenbank wird dieser Tatsache hiermit Rechung getragen.

Es gibt neben der Darstellung **einfacher** und **mehrwertiger** Attribute auch noch die **zusammengesetzten** Attribute. Zur Übung hier ein Beispiel:

```
                    Adresse

                Plz     Ort     Straße
```

Alle Attribute werden mit einer flachen Ellipse umrahmt.

Beziehungen (relationships) werden durch eine Raute dargestellt, die Entitäten natürlich mit einem Rechteck. Zur Übung hier ein Beispiel:

```
          Entität 1 ------ Beziehungstyp ------ Entität 2
```

Hier nocheinmal die Zusammenfassung der Darstellungen, die ab hier im Dokument verwendet werden:

Entitäten und Attribute:

```
                                    |Buch|

          (InvNr)         (Titel)         ((Autor))       (Verlag)        (ISBN)
```

Mehrfache Attribute werden so dargestellt:

```
                    (Adresse)

                (Plz)   (Ort)   (Straße)
```

Beziehungen (relationships) werden nun so dargestellt:

|Entität 1|------<Beziehungstyp>------|Entität 2|

Die Art der Beziehungen der einzelnen Entities entscheidet später über die Struktur der SQL Datenbank, also wie die Attribute geordnet werden, wieviele Tabellen angelegt werden müssen, u.s.w. Hier stellt sich die Frage nach der Komplexität der Beziehungen (relationships). Die Komplexität der Beziehungen läßt sich an einem kleinen Beispiel darstellen:

|Person|-----<wohnt in>-----|Ort|

Dieses Beispiel ist eine 1:1 Beziehung, was bedeutet, daß man in der Datenbank nicht mehrere Felder für den Wohnort einer Person reservieren muß. Die Tatsache, daß dieses eine 1:1 Beziehung ist, wird durch eine kleine 1 in dem Diagramm dargestellt:

1 1

|Person|-----<wohnt in>-----|Ort|

Komplexe Beziehungen stellen eine Entität dar, die zu einer bestimmten Zeit mit mehreren Entitäten in Beziehung steht:

1 N

|Leser|-----<leiht aus>-----|Buch|

Es ist völlig klar, daß ein Leser zu einem bestimmten Zeitpunkt mehrere Bücher ausleihen kann. In einer Bücherei können von einem Buch mehrere Exemplare vorhanden sein, die dann von mehreren Lesern gleichzeitig ausgeliehen werden können. Das ist mit obiger Beziehung aber nicht gemeint. Jedes Buch in einer Bibliothek erhält immer eine eindeutige Inventarnummer. Alle Exemplare eines Buches in einer Bibliothek gehören zu einer Entitätsmenge, die aus mehreren gleichen Entities besteht. Jedes anfassbare Buch erhält in einer Bibliothek eine eindeutige Inventarnummer, genauso, wie der Leser eine eindeutige Mitgliedsnummer erhält. Für die Relation bedeutet dies genaugenommen, daß nur zwischen dem Leser, eindeutig identifiziert durch die Mitgliedsnummer und dem Buch, identifiziert durch eine eindeutige Inventarnummer eine 1:N Beziehung bestehen kann. Im Grunde genommen muß es also heißen:

```
          1                                  N
|Mitglied 523|-----<leiht aus>-----|Inventar 1067|
|Mitglied 523|-----<leiht aus>-----|Inventar 1068|
```

Inventar Nummer 1067 und 1068 können auch 2 Exemplare vom gleichen Buch sein. Wie man sieht ist, es nicht so einfach, die elementaren Dinge korrekt zu erfassen und in Beziehung zueinander zu setzen.

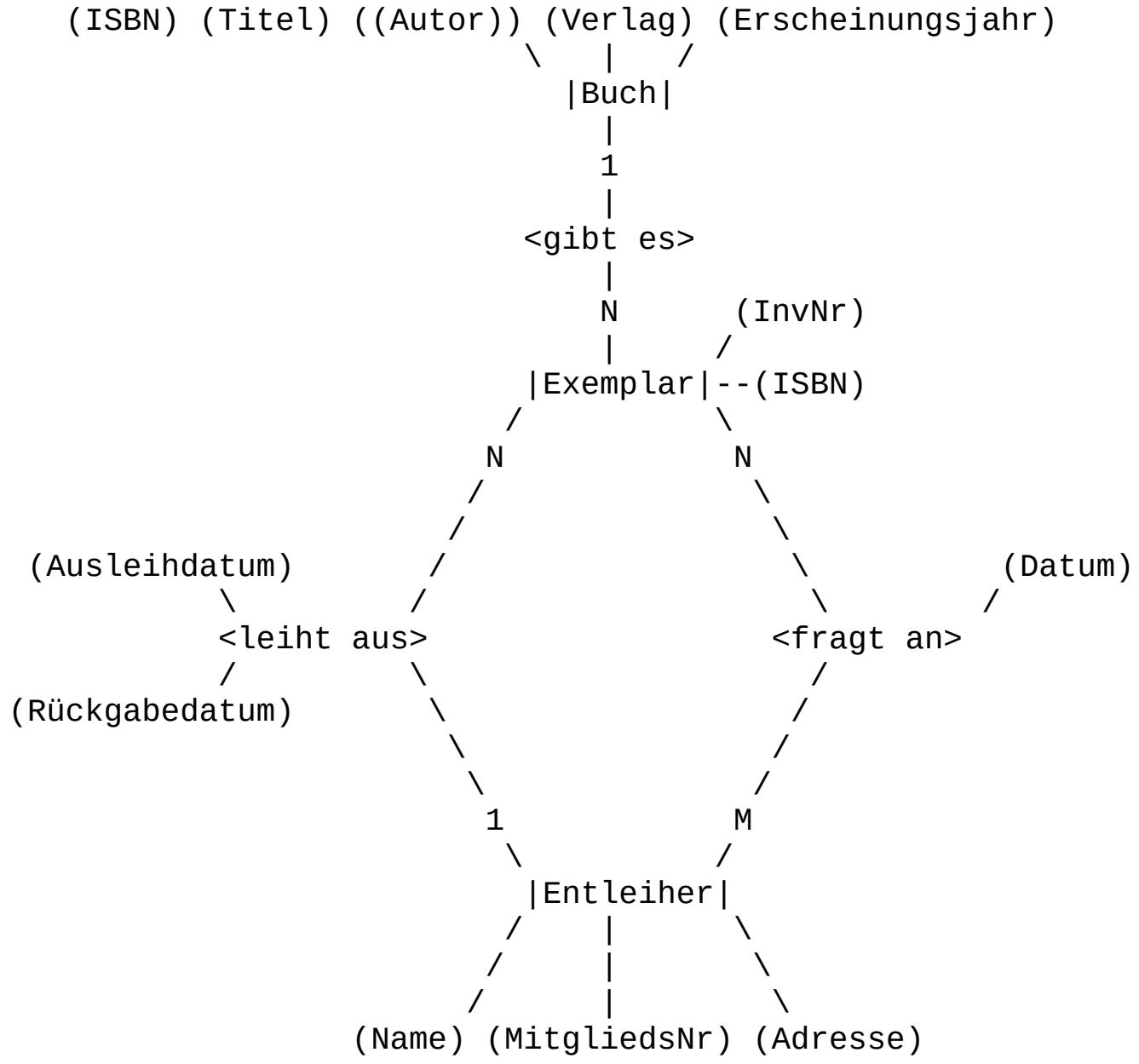


43.2 Wertigkeiten der Relationen

Bei der Darstellung der Entity-Typen muß man genauer zwischen den Wertigkeiten der Beziehungsarten unterscheiden. Insgesamt erhält man 4 verschiedene Beziehungsarten:

1. 1:1 eins zu eins
2. 1:N eins zu viele
3. N:1 viele zu eins
4. N:M viele zu viele

Damit dies etwas klarer wird, hier ein Diagramm als Endergebnis der Planungen. Wie man dahin kommt, wird danach detailliert beschrieben.



Hier kann man nun genau die verschiedenen Beziehungsarten genau sehen. Betrachten wir die Relation zwischen den Entitäten Buch und Exemplar (gibt es). Hier gibt es eine 1:N Beziehung, weil von einem Buch mehrere Exemplare, also eine Entitätsmenge eines Buches in der Bibliothek vorhanden sein kann. Die Relation (leiht aus) stellt zwischen dem Entleiher und den Exemplaren eine 1:N Beziehung, bzw. zwischen den Exemplaren und dem Entleiher eine N:1 Beziehung dar. Komplizierter wird es bei der Relation (fragt an). Hier möchte gerne ein Entleiher mehrere Bücher reservieren lassen. Nun hat die Bibliothek viele Mitglieder (M), die jeweils mehrere Bücher (N) ausleihen oder reservieren können. Hier ist eindeutig eine N:M Relation gegeben.






43.3 Wie entwickelt man denn nun ein solches ER-Diagramm ?

Zuerst besorgt man sich am besten eine Tapetenrolle, auf der man endlos kritzeln, verwerfen und neu aufzeichnen kann. (So machen es viele, wenn Sie es überhaupt machen) Hier kann man sich dann austoben. Es gibt aber auch die Möglichkeit, systematisch ein solches Diagramm zu entwickeln.




43.4 Systematische Entwicklung anhand eines Beispiels

Man sucht sich zuerst alle Entitäten zusammen, also alle Dinge, Lebewesen, Begriffe, Ereignisse und Vorgänge, also Handlungen:

Für unser Beispiel können wir also auflisten:

- Buch
- Person
- Der Vorgang des Ausleihens
- Vorgang der Reservierung (Anfrage)

Nun fragen wir uns nach den Attributen, die ein Buch charakterisieren:

- ISBN-Nummer
- Titel
- Die Autoren
- Verlag
- Erscheinungsjahr

Nun alle Attribute, die eine Person charakterisieren:

- Name
- Adresse
- Anschrift
- ...

Wenn es von Entities, also Buch und Person mehrere gibt, dann müssen wir diesen eindeutige Nummern zuweisen. Dies sind hier die Inventarnummer für das Buch und eine Mitgliedsnummer für die Person. Dieses zusätzliche Attribut hat einen einfachen Sinn. Man kann dann in der Datenbank mit wenig Platz, also nur zwei Spalten einer Mitgliedsnummer eine Buchnummer zuordnen. Man erspart sich damit, daß man immer Name, Adresse, Anschrift..u.s.w. in derselben Tabelle mit abspeichern muß. Wir korrigieren also die Attribute für das Buch:

- ISBN-Nummer
- Titel
- Die Autoren
- Verlag
- Erscheinungsjahr
- **Inventarnummer**

Und für die Person:

- Name

- Adresse
- Anschrift
- **Mitgliedsnummer**

Nun haben wir noch zwei Vorgänge (Entities): Das Ausleihen und die Anfrage. Der Vorgang des Ausleihens und der Anfrage ist an einen Zeitpunkt gebunden. Die Angabe des Datums ist hierfür sinnvoll. Wir ordnen also den beiden Relationen Attribute zu:

Ausleihen: Ausleihdatum, Rückgabedatum

Anfrage/Reservierung: Reservierungsdatum

Wenn man sich nun das Diagramm nochmals anschaut, dann ist klar, daß wird fast alle Entities schon einzeichnen können, wir haben aber zwei Dinge noch nicht berücksichtigt. Zum einen fehlen uns die Wertigkeiten bei den Relationen der Entities untereinander und zum Zweiten ist die Relation zwischen Buch und Exemplar noch nicht klar. Bei Personen ist es klar, jeder Mensch ist einmalig, ein Buch nicht.

Die Entitäten müssen also noch in zwei Kategorien eingeordnet werden können:

1. Einmalige Dinge
2. Nicht einmalige Dinge

Danach untersuchen wir nun unsere Entitäten:

- Personen sind einmalig, von keinem Menschen gibt es zwei Exemplare
- Bücher sind eindeutig nicht einmalig, es gibt eventuell viele Exemplare
- Das Ausleihen ist ebenfalls nicht einmalig, es kann beliebig oft geschehen
- Die Anfrage bzw. die Reservierung eines Buches kann ebenfalls häufiger geschehen

Uns interessieren im Moment jedoch nur die Dinge, die nicht einmalig sind. In unserem Beispiel sind das die Bücher. Wir müssen also noch eine weitere Relation und auch eine neue Entität (Exemplar) einführen, damit Bücher mit Exemplaren verknüpft werden können. Hier ist klar, daß auf ein Buch mehrere Exemplare kommen. Wir haben also hier eine neue 1:N Relation und eine neue Entität.

Daher müssen wir die Attribute für die Entitäten Buch und Exemplar noch einmal neu überdenken:

Bisherige Attribute für Buch:

- ISBN-Nummer
- Titel
- Die Autoren
- Verlag
- Erscheinungsjahr
- **Inventarnummer**

Die Attribute für Buch und Exemplar müssen neu vergeben werden, und zwar nach folgenden Kriterien:

1. Attribute, die ein Buch eindeutig beschreiben
2. Attribute, die ein Exemplar eindeutig festlegen
3. Gemeinsame Attribute, damit ein Exemplar einem Buch eindeutig zugeordnet werden kann

Daraus ergibt sich eindeutig folgende Zuordung der Attribute für das Buch:

- ISBN-Nummer
- Titel
- Die Autoren
- Verlag
- Erscheinungsjahr

Und für die Entität Exemplar:

- Inventarnummer
- ISBN-Nummer

Wir müssen noch die Wertigkeiten bei den Vorgängen zuordnen. Die Anfrage/Reservierung betrifft Personen und Exemplare (nicht Bücher). Hier ist eine N:M Relation gegeben.

Der Vorgang des Ausleihens bezieht sich auch auf Personen und Exemplare, allerdings kann hier nur eine Person mehrere Exemplare ausleihen. Wir berücksichtigen hier nicht, daß es viele Personen gibt, die viele Bücher ausgeliehen haben können, weil wir jeden Vorgang des Ausleihens für sich allein betrachten, und nicht die Summe der Vorgänge.

Wir sind am Ende der Analyse und können nun die Fakten in ein Diagramm einzeichnen. Der Ort, wo die Entitäten, also die Person, das Exemplar, und die beiden Vorgänge eingezeichnet werden, ist zwangsläufig anhand der Relationen gegeben:

Wir haben hier Entleiher und das Buchexemplar, zwischen denen die Vorgänge Ausleihen und Anfragen/Reservieren ablaufen. Einfach, nicht ?

Zwischen Exemplar und Buch gibt es eine eindeutige 1:N Relation, also malen wir die Entität Buch über die Entität Exemplar und dazwischen die Relation.

Wir sind nun am Ende der Entwicklung eines ER - Diagramms. fassen wir noch einmal die Systematik der Entwicklung zusammen:

1. Man sucht sich zuerst alle Entitäten zusammen, also alle Dinge, Lebewesen, Begriffe, Ereignisse und Vorgänge, also alle Handlungen.
2. Es werden die Attribute zu den Entitäten zugeordnet
3. Es werden nun die Entitäten in zwei Kategorien eingeordnet: Einmalig und nicht einmalig, also mehrfach vorhanden
4. Allen einmaligen Entitäten wird ein neues Attribut, also Indentifikationsnummer zugeordnet
5. Alle **nicht** einmalige Entitäten werden in zwei eigene Entitäten aufgespalten und zwar nach folgenden Kriterien:
    1. Attribute, die die (ursprungs) Entität eindeutig beschreiben
    2. Attribute, die ein Exemplar dieser (ursprungs) Entität eindeutig festlegen
    3. Gemeinsame Attribute, damit ein Exemplar der (ursprungs) Entität eindeutig zugeordnet werden kann

    Nicht einmalige Entitäten werden somit in zwei Entitäten aufgespalten, der Entität Exemplar und der ursprünglichen Entität.
6. Zuordnung der Wertigkeiten der Vorgänge oder Ereignisse in Relation zu den Dingen, Lebewesen und Begriffen

7. Festlegung des Ortes beim Einzeichnen in das Diagramm, die sich immer zwangsläufig ergibt.








43.5 Was kann man aus diesem Diagramm nun entnehmen ?

Zuerst dient es dem Datenbankdaministrator und den Entwicklern der Interfaces (ACCESS, PHP3) als Übersicht. Zum zweiten kann man daraus mit ein wenig Erfahrung direkt das CREATE Statement der **Data Definition Language** (DDL) formulieren. Dem ER - Diagramm kann man direkt entnehmen, wieviele unabhängigge Tabellen unser obiges Beispiel zumindest benötigt und welche Attribute darin enthalten sind. Zusätzlich müssen alle Tabellen, die miteinander vernüpft werden müssen und können, zumindest ein gemeinsames Attribut enthalten. Man muß sich daher überlegen, welche Informationen für die SELECT Statements aus den Tabellen entnommen werden müssen, und wie diese eventuell miteinander verknüpft werden können. Hier eine erste Aufspaltung der Daten in einzelne Tabellen:

1. Tabelle **Buch** mit den Attributen ISBN, Titel, Autoren, Verlag, Erscheinungsjahr
2. Tabelle Exemplar mit den Attributen ISBN und Inventarnummer
3. Tabelle Entleiher mit den Attributen Name, Adresse, Mitgliedsnummer
4. Tabelle Ausleihe mit den Attributen Entleihdatum, Rückgabedatum, Mitgliedsnummer und Inventarnummer
5. Tabelle Anfrage/Reservierung mit den Attributen Datum, ISBN-Nummer und Mitgliedsnummer

Für eine Verknüpfung der Tabellen Buch und Exemplar reicht das gemeinsame Attribut ISBN-Nummer. Für die Verknüpfung von Exemplar mit Entleiher über die Relation Ausleihe gibt es die Gemeinsamkeit Inventarnummer und Mitgliedsnummer. Bei der Verknüpfung über die Relation Anfrage/Reservierung dient die Mitgliedsnummer und die ISBN-Nummer. Dies ist eine Besonderheit, die näher betrachtet werden muß. Eine Reservierung eines Entleihers wird ja für ein Buch durchgeführt. Diesem ist dabei völlig egal, welches Exemplar dieses Buches er schließlich erhält. Man kann jedoch über eine weitere Tabellenverknüpfung ermitteln, welches Exemplar eines Buches gerade zurückgegeben worden ist, also im Archiv verfügbar ist. Nun stellt sich die Frage, ob es nicht sinnvoll ist, ein Attribut Entliehen der Tabelle Exemplar hinzuzufügen. Man kann anhand der Tabelle Ausleihe stets ermitteln, welche Bücher ausgeliehen sind und welche wann spätestens zurückgegeben werden. Für eine Reservierung eines Buches benötigt man ein zusätzliches Attribut in der Tabelle Exemplar.

Mit all diesen Informationen aus der Erstellung des ER - Diagramms kann man schon recht zügig und treffsicher die Datenbank planen und aufstellen. Leider muß man bei größeren Datenbanken noch ein paar weitere Prozesse durchführen. Diese werden im folgenden Kapitel behandelt. Viele der im Folgenden beschriebenen Vorgänge haben wir nämlich automatisch richtig gemacht, jedoch ohne uns darüber bewußt gewesen zu sein.







44. Tutorial Normalisierung von Datenbanken

Dieses Kapitel sollten Sie sich unbedingt durchlesen, bevor Sie anfangen die Struktur der MySQL Datenbank festzulegen. Dieses Kapitel ist insbesondere für Einsteiger gedacht, die anhand der Fehler von Obsthändler Krämer mit seinen für Anfänger typischen Fehlern (Herr Krämer hat zuvor wirklich noch nichts mit Datenbanken zu tun gehabt...) lernen können, wie man eine Datenbank richtig plant.







44.1 Sinn und Zweck der Normalisierung

Bei komplexen Systemen mit vielen Objekttypen und Beziehungen ist die Modellierung der Datenstruktur nicht unproblematisch. Hierzu wurden Regeln aufgestellt, nach denen man die Einträge in den Tabellen ordnet und aufteilt. Die wichtigsten Ziele der Normalisierung einer oder mehreren Tabellen sind:

- Vermeidung von Redundanzen (doppelt gespeicherten Werten)
- Vereinfachung des Aufbaues von Tabellen
- Einfache Datenpflege







44.2 Obsthändler Krämers erster Datenbankentwurf

Damit dieses ein wenig klarer wird, möchte ich anhand eines Beispiels einmal darstellen, wie eine völlig unerfahrene Person eine Datenbank planen würde. Ein Obsthändler Krämer hat folgenden Datenbankentwurf erstellt:

Auftragsnr.	Datum	Kunde	Artikelnr.	Bezeichnung	Menge
1	1.1.99	1 Schmitt, Bonn	134	Coxorange	4 Kisten
1	1.1.99	1 Schmitt, Bonn	135	Kiwi	4 Kisten
2	1.1.99	2 Müller, Köln	140	Butterbirne	2 Kisten
2	1.1.99	2 Müller, Köln	160	Kürbis, rot	2 Stück
2	1.1.99	2 Müller, Köln	160	Kürbis, gelb	10 Stück
3	2.2.99	1 Schmitz, Bonn	103	Johannisbeeren	5 Kilo
3	2.2.99	1 Schmitz, Bonn	134	Coxorange	12 Kisten
3	2.2.99	1 Schmitz, Bonn	135	Kiwi	2 Kisten
4	2.2.99	45 Lehmann, Jülich	30	Bananen	12 Kilo
4	2.2.99	45 Lehmenn, Jülich	27	Ananas	60 Stück

Herr Krämer hat die Datenbank nach seiner Intuition mit ACCESS erstellt. Wie man sehen kann, hat er die Daten so eingegeben, wie er es seit vielen Jahren auf dem Papier auch tut. Im Prinzip tuts dieser Datenbankentwurf auch, nur leider gibt es hier viele Probleme:

1. Viele gleiche Einträge (Redundanz) sorgen nach ein paar Monaten dafür, daß die Datenbank unnötig groß wird.
2. Tippfehler (letzte Zeile: Lehmenn) machen das Auffinden aller Lieferungen an Lehmann, Jülich unmöglich.
3. In der Spalte Kunde sind sowohl Kundennummer, Name und Ort zugleich eingetragen. Herr Krämer muß sich stets selber die Kundennummer merken und alle Daten bei jedem neuen Auftrag stets neu eintippen.
4. Artikelnummer und Bezeichnung könnten als Synonyme verwendet werden, sind aber hier jeweils für sich in getrennten Spalten gespeichert.
5. Die Farbe des Kürbis gibt eine andere Art an, die eine neue Artikelnummer erfordert.
6. Die Spalte Menge enthält sowohl die Stückzahl als auch die Einheit der Ware (Kiste, Stück, Kilo). Die Einheit ist aber stets an die Art der Ware gekoppelt.

Wenn man sich noch einmal die wichtigsten Ziele der Normalisierung betrachtet, hatt Herr Krämer noch einige Arbeit vor sich, die wir nun zunächst einmal etwas theoretischer abhandeln möchten.







44.3 Definition der Normalformen

Die Theorie der Normalisierung, die durch E.F. Codd begründet worden ist, besteht aus 9 Regeln, die auch als 1. - 9. Normalform bezeichnet werden, von denen die ersten 3 die wichtigsten sind. Man versetzt die Datenbank jeweils schrittweise in die Zustände der 1. - 9. Normalenform. Hier die Definitionen der ersten 5 am häufigsten verwendeten Normalformen:

1. Normalform: Eine Relation befindet sich in der ersten Normalform, wenn keines ihrer Attribute eine untergeordnete Relation darstellt und wenn alle Attribute nur atomare Werte beinhalten.
2. Normalform: Laut Definition muß die Datenbank immer zuerst in die erste Normalform versetzt werden, bevor man diese in die 2. Normalform versetzen kann. Hierbei müssen alle nicht zum Schlüssel gehörenden Attribute von diesem voll funktional abhängig sein. Besteht ein Schlüssel aus mehreren Teilschlüsseln, so ist das Element aus dem Datensatz herauszuziehen, welches nur von einem Teilschlüssel abhängt.
3. Normalform: Zusätzlich zur 2. Normalform gilt für jeden Schlüssel: Alle nicht zum Schlüssel gehörende Attribute sind nicht von diesem transitiv abhängig. Das bedeutet, daß alle Attribute nur vom Schlüsselattribut, nicht aber von anderen Attributen abhängig sein. Eine Abhängigkeit zwischen den Attributen muß aufgelöst werden.
4. Normalform: Diese 4. Normalform betrifft Mehrfachabhängigkeiten von Attributmengen von einem Superschlüssel (Übergeordneten Schlüssel). Diese muß in Einzelabhängigkeiten (Siehe 3. NF) aufgelöst werden.
5. Normalform: Falls in der 4. Normalform keine verlustlose Zerlegung in Einzelabhängigkeiten möglich ist, dann muß eventuell 2., 3. ... Superschlüssel hinzugezogen werden, solange, bis nur noch Einzelabhängigkeiten der Attribute von einem oder mehreren Superschlüsseln bestehen.







44.4 Anomalien

Mit dem Versetzen einer Datenbank in eine Normalform möchte man eine ganze Reihe von seltsamen Effekten vermeiden, die Anomalien genannt werden. Hier gibt es z.B. die Einfüge-Anomalie. Geht man von der auch als Universalschema oder Universalrelation genannte Tabelle aus, in der alle Attribute, Entitäten in einer Tabelle gespeichert werden (ähnlich Herrn Krämers Datenbankentwurf) aus, so können folgende Anomalien auftreten:

1. Einfüge-Anomalie: Ein Lieferant/Kunde kann nur dann eingetragen werden, wenn er auch eine Ware liefert/kauft. Das ist bei Herrn Krämers Datenbankentwurf der Fall.
2. Lösch-Anomalie: Wird die einzige Lieferung an einen Kunden/von einem Lieferanten gelöscht, weil diese nicht angekommen ist, oder weil dieser eine Ware nicht mehr liefert, so sind auch die Informationen über Wohnort, Anschrift ... verloren. Auch dies trifft auf unser Beispiel zu.
3. Änderungs-Anomalie: Falls ein Lieferant/Kunde umzieht, so sind mehrere Tupel (Einträge in die Datenbank) eventuell nachträglich zu ändern, da ansonsten die Rechnungen sowohl an die alte sowie auch an die neue Anschrift adressiert werden.
4. Update-Anomalie: Wird eine Korrektur z.B. bei einer Artikelnummer durchgeführt, so kann es passieren, daß entweder alle bisherigen Einträge ebenfalls aktualisiert werden müssen, oder daß es zu Inkonsistenzen führt. Dabei bezeichnet dann eine Artikelnummer gleich mehrere Sorten von Obst innerhalb der Datenbank.




44.5 Praxisanwendung Normalisierung

Wir werden nun Herrn Krämers Datenbankentwurf der Prozedur der Normalisierung unterwerfen.

Hier noch einmal der ursprüngliche Datenbankentwurf:

```
Auftragsnr.     Datum   Kunde                   Artikelnr.      Bezeichnung     Menge
1               1.1.99  1 Schmitt, Bonn         134             Coxorange       4
Kisten
....
4               2.2.99  45 Lehmenn, Jülich      27              Ananas          60
Stück
```

Hier nun die Datenbank in der 1. Normalform:

```
AuftrNr.        Datum   KundenNr.       Name    Ort     ArtNr.  Bez.    Menge
```

Es wurde in der Spalte **Kunde** die Abhängigkeit von Kundennummer, Name und Ort aufgelöst. Dafür sind weitere Spalten entstanden. Nach der Trennung entspricht der Forderung nach atomaren Werten (von atomos = unteilbar).

Nun folgt die 2. Normalform:

```
1. Tabelle:     AuftrNr.        Datum   KundenNr.       Name    Ort

2. Tabelle:     AuftrNr.        ArtNr.  Bez.            Menge
```

Es wurde die Datenbank in zwei Teile aufgeteilt. Der Grund liegt darin, daß hier verschiedene, unerlaubte Abhängigkeiten in einer Tabelle enthalten sind. Zum einen gehören Kundennummer, Name, Ort (und Anschrift) und Artikelnummer, Bezeichnung und Menge jeweils zusammen. Die Auftragsnummer in der 1. Tabelle ist der sogenannte Primärschlüssel, die Auftragsnummer in der 2. Tabelle ist der Fremdschlüssel. Wir müssen diese beiden Tabellen durch eine Relation miteinander vernüpft betrachten: Kunde x (kauft) Ware y (Siehe ER-Diagramm). Der Vorgang wird mit Kauf bezeichnet und erhält eine laufende Nummer, damit man diese später sauber voneinander trennen kann. Diese laufende Nummer ist die Auftragsnummer, die in beiden Tabellen als Bindeglied enthalten sein muß. Ohne Bindeglied kann man die Waren nicht mehr dem Käufer zuordnen. Die Forderung, daß alle Attribute von dem Schlüssel voll funkional abhängig sein sollten, konnte nur dadurch erfüllt werden, daß die Tabelle in 2 Tabellen aufgeteilt wurde, bei denen die Attribute von ihrem Schlüssel funktional voll abhängig sind. Hierbei war dann der Schlüssel der einen Tabelle auch in der anderen verwendbar, nämlich als Fremdschlüssel.

Die Dritte Normalform ist etwas komplexer:

```
1. Tabelle:     ArtNr.          Bez.

2. Tabelle:     AuftrNr.        Menge           ArtNr.

3. Tabelle:     AuftrNr.        KundenNr.       Datum

4. Tabelle:     KundenNr.       Name    Ort     (Anschrift...)
```

Bei dem Versetzen der Datenbank in die 3. Normalform wurde die Forderung nach der Abhängigkeit der Attribute nur vom Schlüsselattribut erfüllt. Es durften zwischen den Attributen keinerlei Abhängigkeiten geben. So ist es z.B. nicht erlaubt gewesen, daß der Name des Kunden von der Artikelnummer abhängig gespeichert wird, oder daß Ort und Datum immer zusammen abgelegt werden. Stattdessen wurden nur die wirklich zusammengehörenden Attribute, wie Kundennummer, Name, Ort, (Anschrift) in einer Tabelle gespeichert, und eines dieser Attribute als Schlüsselattribut definiert. Hier ist es die Kundennummer, die als eindeutiger Schlüssel die Verknüpfung zur Tabelle 3 herstellen kann. Damit also alle Attribute in 4 Tabellen miteinander abfragbar werden, müssen zumindest 2 Tabellen jeweils über einen Primärschlüssel und einen Fremdschlüssel verfügen, während die restlichen beiden nur über einen Primärschlüssel verfügen müssen. Hier nun obige Tabellendefinitionen mit der Angebe der Primärschlüssel (PS) und Fremdschlüssel (FS):

```
1. Tabelle:     ArtNr.(PS)      Bez.

2. Tabelle:     AuftrNr.(PS)    Menge           ArtNr.(FS)
```

```
3. Tabelle:     AuftrNr.(PS)    KundenNr.(FS)   Datum

4. Tabelle:     KundenNr.(PS)   Name    Ort     (Anschrift...)
```

Anhand dieses Beispiels **Herrn Krämers** haben Sie nun gelernt, wie man Datenbanken von Anfang an so strukturiert, daß die gefürchteten Anomalien nicht eintreten können. Nun können Sie z.B. mit dem Tutorial Einsteiger Tutorial LINUX MySQL Server fortfahren.





45. Lexikon der Datenbanktechnik

- **ad hoc query** ist eine temporäre Abfrage, bei welcher das Abfragestatement, noch die Ausgabe zwischengespeichert wird.
- Eine **Änderungs - Anomalie** ist einfach anhand eines Beispiels zu beschreiben: Es zieht ein Kunde um. Hierbei müssen mehrere Adressinformationen gleichzeitig geändert werden. Bei gleichzeitigem lesendem Zugriff auf die Datenbank von mehreren Kundenbaratern kann es passieren, daß zwar schon die Anschrift geändert wurde, jedoch noch nicht die Telefonnummer. Ein Locking-Problem.
- "anomalie">Eine **Anomalie** wird gewöhnlich in Untergrupen aufgeteilt: Einfüge - Anomalie, Lösch - Anomalie, Änderungs - Anomalie. Diese können zu Datenverlusten oder zu groben Fehlern führen.
- **Attribut** beschreibt einen Wert, der in jedem Tupel einer Relation vorkommt. Normalerweise wird hiermit eine Spalte einer Relation bezeichnet. Ein Attribut ist eine Eigenschaft einer Entity. Bei ACCESS wird statt des Begriffes Attribut auch der Ausdruck Feld verwendet.
- **base query** Siehe ground query
- **BCNF**Boyce-Codd Normal Form beschreibt bei einem relationalen Modell mehrere Normalformen.
- **Boyce-Codd Abhängigkeiten** beschreiben einen Teil eines Schlüssels, der von einem Attribut abhängig ist, welches nicht zu einem anderen Schlüssel gehört.
- **Boyce-Codd Normalform** beschreibt den Zustand einer Datenbank in der dritten Normalform, bei welcher keine Boyce-Codd Abhängigkeiten mehr enthalten sind.
- **candidate key** stellt ein oder mehrere Attribute dar, welche ein Tupel in einer Relation eindeutig abbilden können. Dies ist ein potentieller **Primary Key**.
- **commit** beschreibt eine Entscheidung, mit einer Änderung in der Datenbank fortzufahren.
- **Errechnetes Attribut** beschreibt ein Attribut, welches sich aus mehreren Werten anderer Attribute errechnet. Diese können in Relationen gespeichert werden. Sie werden so angelegt, wie sie bei einer Abfrage gebraucht werden.
- Eine **Datenbank** ist eine Sammlung aller Entitys und Attribute. Sie beschreibt die Gesammtheit aller verfügbaren Informationen.
- **Datenbank Manager** ist eine Person, die für das Design, die Konstruktion und Wartung einer Datenbank verantwortlich ist. Darunter kann man aber auch ein Stück Software verstehen, mit welchem man die Datenbank administrieren kann.
- **Deadlock** tritt immer dann auf, wenn zwei oder mehrere konkurrierende Prozesse nicht beendet werden können, weil diese darauf warten, daß der jeweils andere Prozeß Resourcen freigibt, die der andere zum Lesen oder Schreiben gesperrt hat. Bei Datenbanken, deren Lockmechanismen (ROW/COLUMN/CELL/TABLE LOCKING) zu grob differenziert sind, entstehen hierdurch oft lange Reaktionszeiten. Häufiger ist allerdings falsche Programmierung die Ursache.
- **Dreiwertigkeit** Unter SQL können Werte in einer Tabelle im Prinzip nicht nur drei Werte

annehmen (0 oder irgendein Wert), sondern auch den Wert **NULL**, was für Daten nicht vorhanden steht.

- **Verteilte Datenbanken** sind Datenbanken, deren Inhalte auf mehrere Computersysteme verteilt sind.
- **Domain** ist eine Sammlung von allen möglichen Werten eines Attributes. Diese kann endlich und unendlich sein. In einer Datenbank existieren Funktionen, die die verschiedenen gespeicherten Werte zählen können.
- **Entity** Eine Entity ist ein existierendes Objekt, eine Beobachtung, eine Transaktion oder Person, über welches Informationen in einer Datenbank abgelegt werden.
- **Entity Relationship (ER) Diagramm** Ein solches Diagramm ist für die Entwurfsphase einer Datenbank unentbehrlich. Aus einem ER - Diagramm wird das Relationenmodell entwickelt. Wer einmal Matrizen in Mathematik behandelt hat, dem wird der Begriff Normalisierung einer Matrix (Normalenform) etwas sagen.
- **Feld** ist ein Ausdruck, der von ACCESS als Synonym für ein Attribut verwendet wird.
- **File** ist eine (von der Datenbank unabhängig) benannte Datei auf dem Filesystem, in welcher eine Relation oder eine ganze Datenbank gespeichert wird.
- **Filter** ist ein Ausdruck für eine dauernde Abfrage eines Datenflusses.
- **Foreign Key** werden eingesetzt, um automatische Abhängigkeiten zwischen Entitys in mehreren Tabellen zu schaffen. Eine Änderung einer Entity einer Tabelle wird dann auch automatisch auf andere Tabellen (Relations) übertragen. Dies führt oft zu diffusen Problemen bei dem Locking.
- **Erste Normalform** Eine Datenbank befindet sich in der ersten Normalform, wenn es keine zusammengesetzen Fehler mehr gibt. Hierbei kann es sich um die Trennung von zusammengehörigen Attributen, wie z.B. Adresse, Postleitzahl, Telefonnummer, Vorwahl... handeln, als auch um die Trennung von Wiederholfeldern, wie z.B. einer Auflistung von entliehenen Büchern oder anderen Ereignissen.
- **Granularität** beschreibt die Größe der kleinsten Einheit, die unabhängig gelockt werden kann. Eine Datenbank kann Locks (Lese/Schreibsperren) auf Datenbank/Relation/Tupel/Attribut - Ebene setzen. Es sind auch Locks von einzelnen Attributen innerhalb eines Tupels erlaubt. Die Fähigkeiten schwanken je nach Datenbankhersteller.
- **ground query** ist eine Abfrage, bei der alle Attribute direkt aus den Relationen entnommen werden. Es werden hierbei keine Ergebnisse anderer Abfragen verwendet. Diese wird auch ref id="basequery" name="**base query**"> genannt.
- **Hash-Tables**Hash-Tables sind Zahlen, die sich z.B. aus dem Namen einer Person, bzw. der Buchstaben errechnet. Hierzu werden die ASCII Werte addiert, Modulo einer Primzahl, die größer als die Zahl der Einträge in der Datenbank ist. Diese Hash-Zahl beschreibt zumeist eindeutig den Ort des Eintrags in der Datenbank. Er ermöglicht das schnelle Auffinden von Einträgen. Typischerweise beträgt die Größe eines HASH-Table ca. 5-10% der Datenbank.
- **Identität** ist eine Relation innerhalb von Attributen
- Ein **INDEX** ist eine Methode, um Tupel in der Datenbank neu zu ordnen, oder diese in spezieller Reihenfolge bei der Ausgabe zu ordnen. Indizes werden auch dazu verwandt, um die Abfrage stark zu beschleunigen. In der Datenbank werden hierzu intern ref id="hash" name="Hash-Tables"> angelegt.
- **Inversion** ist ein Index, der die umgekehrte Suche nach Einträgen in der Datenbank zuläßt.

- **Join** ist eine Abfrage, welche Daten mehrerer Relationen verwendet und miteinander verknüpft. Die Relationen müssen zumindest ein Attribut (auch join- oder linking Attribut genannt) gemeinsam haben.
- **Key / Schlüssel** ist ein Attribut oder eine Kombination von Attributen. Eine Kombination ihrer Werte kann dazu verwendet werden, um Tupel einer Relation zu suchen.
- **Locking** ist die Reservierung einer Datenbank, Relation, Tupel oder auch deren Kombination für einen User. Es gibt Schreib- und Lese Locks. Diese Verfahren werden benötigt, um Inkonsistenzen der Datenbankinhalte zu vermeiden, z.B. wenn Clients gleichzeitig versuchen, einen Datensatz zu lesen, der gerade von anderen Personen aktualisiert wird.
- **many-to-many relationship** Ein oder mehrere Tupel einer Relation können durch ein oder mehrere Tupel einer weiteren Relation verbunden werden, indem man einen gemeinsamen Wert eines sog. JOIN Attribut einführt. Das bedeutet, daß jeder Wert des JOIN Attributes viele Male in jeder der Relationen enthalen sein kann.
- **Negation** ist eine Abfrage, die alle Tupel, außer denjenigen, für die eine bestimmter Wert eines Attributs angegeben wurde, ausgibt. Solche Abfragen sind oft langwierig, sodaß viele Datenbanken solche Abfragen nicht unterstützen.
- **Normalform** dient dazu, mehrfache Datensätze und Abhängigkeiten zu vermeiden, den Speicherverbrauch zu reduzieren, und die Performance der Datenbank zu verbessern. Hier zu gibt es genau definierte Regeln, die vorschreiben, wie man eine Datenbank in **normal form** bringt. Siehe auch Kapitel Normalformen
- **one-to-many relationship** Exakt ein Tupel in einer Relation ist über eine gemeinsames JOIN Attribut mit mehreren Tupeln einer anderen Relation verbunden. Das bedeutet, daß jeder Wert der ersten Relation, aber nicht zwangsläufig auch in der zweiten Relation eindeutig ist.
- **one-to-one relationship** Exakt ein Tupel in der einen Relation ist genau einem Tupel in einer anderen Relation zugeordnet. Jeder Wert des JOIN Attributes ist eindeutig einem Wert in jeder der beiden Relationen zugeordnet. Dies kann z.B. eine laufende Nummer sein. (autoincrement)
- **Teilweise Abhängigkeit (partial dependancy)** ist erfüllt, wenn ein Wert eines Attributes (ohne Schlüssel) nur von einem Teil des Schlüssels abhängig ist.
- **persistent query** ist die Speicherung einer Abfrage für eine erneute Abfrage. In Internet Suchmaschinen kann man sich vorstellen, daß Millionen User nach dem Eintrag SEX suchen. Da die Ausgabe stets identisch ist, wird die Abfrage und das Ergebnis zwischengespeichert.
- **Post** ist eine Anweisung, Daten zu verändern. Post kann entweder direkt durchgeführt werden, oder muß solange zurückgestellt werden, bis alle verwandten Änderungen ebenfalls durchgeführt werden können. Siehe auch Commit
- **Primary Key** ist ein Schlüssel, der eindeutig jedes Tupel in einer Relation identifizieren kann. Eine Relation darf nicht mehr als einen Primary Key besitzen.
- **Primary Memory** ist zumeist das RAM, in welchem die Daten abgelegt werden. Unter **MySQL** gibt es sog. temporäre Tabellen, die die Abfrage erheblich beschleunigen, da diese im RAM abgelegt werden. Ein Teil der Selbstoptimierung von **MySQL**.
- **Project** ist eine Abfrage, in welcher nur einige der Attribute der Relation ausgegeben werden.
- **QBE** Query-By-Example ist eine Art Makrosprache für Abfragen.
- **Query** ist eine Abfrage in einer eigenen Sprache. Oft wird hiermit auch die Ausgabe gemeint.
- **Query Resolution** ist der Prozeß des Sammelns von Daten, die für die Ausgabe benötigt werden.

- **Read Lock** ist eine Sperre, die es Clients erlaubt, Daten zu lesen, jedoch nicht zu schreiben.
- **Record** ist ein Synonym für ref id="tuple" name="Tupel">
- **recursive query** ist eine Abfrage, deren Ausgabe dann als Eingabe für dieselbe Abfrage verwendet wird.
- **Relation** ist die Basis einer relationalen Datenbank. In ACCESS Sprache wird hiermit eine Tabelle bezeichnet. Eine Relation enthält normalerweise eine Entität (entity) und deren Attribute (zugehörige, abhängige Reihen oder Tupel). Normalerweise wird die Datenbank in einer Art 2 dimensionalem Koordinatensystem abgespeichert, wobei jede Reihe eine Sammlung von Daten einem Entity-Set darstellt. Die Darstellung aller Entitären und Attribute in einer Relation (Tabelle) wird Universalrelation genannt.
- **Reihe (row)** ist bei der Datenbank Access der Begriff für ein Tupel in der ordentlichen SQL Welt.
- **safe query**Eine Anfrage, bei welcher die Ausgabe endlich ist....
- **schema** ist eine Beschreibung einer Datenbank. Es definiert unter anderem die Relationen, deren Attribute und die Domains der Attribute. In einigen Datenbanken sind **join attributes** definiert als Teil des schema. Ein schema kann auch die Beschreibung einer Relation sein.
- **secondary memory** ist normalerweise die Harddisk oder eventuell auch optische Disks.
- **Zweite Normalform** beschreibt eine Tabelle, in der alle Attribute vom Primärschlüssel abhängig sind. Für den Fall, daß mit zusammengesetzten Primärschlüsseln gearbeitet wird (SQL Befehl INDEX), dann kann ein Attribut von einem Teil des Primärschlüssels abhängen, jedoch ohne von anderen Teilen abhängig zu sein.
- **Sekundärschlüssel (secondary key)** ist neben dem Primärschlüssel ein weitere Schlüssel für eine Relation. Hiermit können auch Fragen nach anderen Kriterien schnell durchgeführt werden, da die Datenbank hierfür intern eine weitere ref id="hash" name="HASH-Tabelle"> anlegt.
- **Select** ist eine Abfrage, in der nur einige Tupel einer Relation ausgegeben werden.
- **SQL** = Sequential Query Language
- **Stored Procedures** stellen eine Art Makro mit Parameterübergabe dar. Man kann somit neue Befehle erzeugen und diese entweder automatisch oder durch Trigger starten lassen. Es erleichtert die Bedienung für Personen, die sich nur wenig mit SQL auskennen. Unter **MySQL** gibt es keine **stored procedures**, es ist aber möglich, Scripte auf der Festplatte zu speichern und diese z.B. Zeitgesteuert ablaufen zu lassen. Siehe Kapitel Batch Modus. In **MySQL** kann man aber einfach eigene Prozeduren in C++ schreiben.
- **Tabelle (table)** ist in ACCESS ein Synonym für Relation
- **Dritte Normalform** beschreibt eine Form, bei welcher kein Attribut von einem Primärschlüssel abhängig ist. Hierzu werden Abhängigkeiten noch weiter aufgelöst, wobei Tabellen weiter in Untertabellen zerlegt werden. Oft muß man hierfür mit weiteren, oft numerischen Attributen arbeiten, um die Abhängigkeiten alle wieder herzustellen. Mathematisch gesehen entspricht dieses Vorgehen in eine Aufteilung in Untermatritzen, die Über einenen Vektor miteinander verknüpft sind (Kreuzprodukt).
- **Transaktion** ist eine Änderung im Datenbestand der Datenbank. Eine einzige Transaktion vermag direkt oder indirekt über Foreign Keys Änderungen in mehreren Relationen oder Tabellen durchführen.
- **Transitive Abhängigkeit (tr. dependency)** ist dann gegeben, wenn der Wert eines

(nicht-Schlüssel) Attributes abhängig von dem Wert einer oder mehrerer (nicht-Schlüssel) Attribute ist. Siehe auch Foreign Key.

- **Trigger** starten Prozeduren oder Abläufe in Abhängigkeit von Zeit, Zellenwert, ... Bei großen DATA WARE HOUSING Lösungen werden viele Prozesse über Trigger in Gang gesetzt.
- **Tupel**In einer Relation wird die Sammlung aller Fakten zu einer Einheit zusammengefasst. Das bedeutet z.B., daß Name, Adresse, Postleitzahl...in einer Spalte zusammengefasst werden. In der Ersten Normalform werden diese Tupel auf mehrere Rlationen (Tabellen) verteilt.
- **two-phase commmit** ist eine Strategie, in welcher Eintrage in die Datenbank oder Änderungen zeitweilig in einer temporären Version der Datenbank durchgeführt werden. Wird dann festgestellt, daß alle Änderungen ohne Probleme durchgeführt werden können, werden diese in die echte Datenbank geschrieben. Unter NOVELL gibt es das sogenannte TTS, welches auf Datei-Ebene so etwas unterstützt. Bei einem Wechsel auf andere Betriebssysteme kann dies zu Problemen führen.
- **Universalrelation** oder auch **Universalschema** ist die Darstellung aller Entitäten, Attribute, u.s.w. ein einer Relation (Tabelle). Diese Darstellung ist typisch für den ersten Entwurf (Sammlung von Fakten) und die Vorbereitung für die Normalisierung. Setzt man eine solche Relation tatächlich ein, dann birt dies die Gefahr, daß bestimmte Anomalien auftreten können, die zu logischen Fehlern und Datenverlusten führen können.
- **unsafe query** ist eine Abfrage, bei der die Ausgabe möglicherweise unendlich ist. Dies trifft häufig auf Rekursive Abfragen zu, die Negation unterstützen. Diese sollten prinzipiell verboten werden.
- **Value** ist der interne Wert eines Wertes einer Entity (z.B. NULL)
- **write lock** ist eine Sperre, die es dem User, dem die Sperre zugeteilt wurde, erlaubt, exklusiv in die entsprechenden Datensätze zu schreiben. Andere User werden solange gesperrt.






46. Verteilte Datenbanken

Verteilte Datenbanken mit MySQL sind recht einfach zu realisieren, wenn man einmal verstanden hat, worum es eigendlich genau geht, welche Probleme auftreten können, und wie diese zu lösen sind. Hier werden nun einige (u.a. auch von mir) erfolgreich implementierte Konzepte zu verteilten Datenbanken vorgestellt:







46.1 Datenbank Spiegelung im Intranet

Die Spiegelung einer **MySQL** Datenbank im Intranet kann z.B. für Backup´s der Datenbank, sinnvoll sein. Hier hat man im Prinzip zwei Möglichkeiten. Zum einen kann man die Festplatten zweier Server über SAMBA oder FTP miteinander verbinden und zeitlich gesteuert über einfache Shellscripte die Datenbank auf den anderen Server überspielen. Zuvor muß jedoch sichergestellt sein, daß keine Schreib/Lese Zugriffe Inkonsistenzen erzeugen. Hierzu geht man Schrittweise vor:

1. Dem MySQL-Client im Batch Modus über den Befehl Grant die Schreib-und Leserechte entziehen.
2. Kopieren der drei Datenbankdateien von MySQL (Daten, Index - und Konfigurationsdatei) über das Filesystem oder via FTP auf das Zielsystem
3. Mit GRANT auf dem Ausgangsserver die Schreib/Leserechte wiederherstellen
4. Neustart des **mysqld** auf dem Zielsystem.

Eine weitere Variante ist der Einsatz des **CODA** Filesystems unter LINUX. Das CODA Filesystem ist in allen neueren Kerneln von LINUX enthalten und sorgt dafür, daß Dateien auf allen eingebundenen Servern auf demselben Stand sind. Hierzu wird bei jeder Änderung in einem File diese auch den anderen im CODA-Filesystem eingebundenen Server diese Änderung differenziell mitgeteilt. Es wird also nicht die ganze Datei bei jeder Änderung kopiert, sondern immer nur die relavanten Teile. Das CODA Filesystem ist vergleichbar mit dem kommerziellen Andrew File System (AFS) von Transarc, welches seit Jahren in großen Rechenzentren zum Einsatz kommt. Bei dieser Variante kümmert sich der Anwender um nichts, es wird alles vom CODA Filesystem geregelt.

Im Intranet braucht man sich im Allgemeinen nicht weiter um die zur Verfügung stehende Bandbreite kümmern, da zumeist genug Bandbreite verfügbar ist.







46.2 Lastverteilung von SQL-Datenbanken im Intranet

Lastverteilung beinhaltet stets, daß die Datenbanken beide auf demselben Stand gehalten werden. Es müssen also bei Schreibvorgängen Änderungen, Löschungen u.s.w. stets bei beiden Datenbanken simultan durchgeführt werden. Lesezugriffe hingegen können auf eine der Datenbanken erfolgen. Um das Problem mit den Schreibzugriffen zu lösen, kann man sich wie bei obigem Beispiel auf das Coda-Filesystem unter LINUX verlassen, oder man ändert die Benutzterschnittestellen (WWW-Interfaces, ACCESS) dahingehend um, daß sämtliche Änderungen in beide Datenbanken zugleich geschrieben werden. Da beim Ausfall einer Datenbank mit Fehlern zu rechnen ist, sollte man von dieser Lösung abstand nehmen, und das die Variante mit dem CODA-Filesystem bevorzugen. Zur Lastverteilung (load balancing) beim Lesen der Daten gibt es auf der Site http://www.freshmeat.net eine Fülle von Software, aus der man je nach Anforderungen auswählen kann.







46.3 Lastverteilung von SQL-Datenbanken im Internet

Im Internet sind die Anforderungen bezüglich der Gleichzeitigkeit, der Geschwindigkeit und der Sicherheit ungleich höher, als im Intranet. Hier bietet sich zur Lastverteilung und zum Schutz vor DoS Angriffen die SINUS Firewall-1 an. Danke der flexiblen Programmiersprache läßt sich mit wenige Aufwand eine einfache Lastverteilung über Port-Redirection realisieren.







46.4 Verteilte Datenbanken mit Standleitung

Standleitungen besitzen zumeist eine recht geringe Bandbreite. Es spricht im Prinzip nichts dagegen, mit Hilfe des Coda Filesystems zu arbeiten. Man sollte jedoch mit Hilfe der TOS - Flags im LINUX Kernel 2.2 die Priorität der IP Pakete, die über die Standleitung laufen, herabsetzen.







46.5 Verteilte Datenbanken mit DIAL-ON-DEMAND Leitung

Die Anforderungen bei verteilten Datenbanken mit einer DIAL-ON-DEMAND Leitung sind völlig andere, als bisher beschrieben. Die Synchronisation der Daten muß schnell und zügig geschehen. Hierzu muß man wissen, welche Daten sich geändert haben, also welche Datensätze gelöscht worden sind, welche hinzugekommen sind, welche verändert wurden. Hierzu muß man ein Journal über das Datum und die Uhrzeit sämtlicher Schreib-und Änderungsvorgänge der Datenbank anlegen. MySQL kann dieses Problem aber mit Hilfe eines Workarounds umgehen. Hierzu fügt man eine weitere Tabelle vom Typ **TIMESTAMP** in die Datenbank ein, die Datum und Uhrzeit jeder Änderung mitführt. Hierzu muß nicht in die Struktur der Statements eingegriffen werden. Sollen nun alle Änderungen an eine Master-Datenbank via Internet oder Direktverbindung übermittelt werden, so muß man mit Hilfe des Batch-Modus folgende Schritte durchführen:

1. Schreibzugriffe mit Hilfe von GRANT unterbinden
2. Mit Hilfe eines auf die Datenbankstrukturen angepasstes Batch Scriptes werden alle Datensätze, die sich im FELD mit TIMESTAMP seit dem letzten Abgleich verändert haben, in eine ASCII-Datei im MySQL Verzeichnis exportiert.
3. Die ASCII-Daten werden Hilfe von z.B. ZIP komprimiert
4. Danach kann die Übertragung via FTP über das Internet erfolgen.

Soweit zum Client, nun aber zur Master Datenbank:

1. Auf dem Zielserver wird wiederum ein Batch-Script ausgeführt, welches die ASCII Datein mit den Änderungen dekomprimiert.
2. Die ASCII-Daten werden wird im Batchmodus in eine zuvor neu angelegte oder geleerte Tabellen in der Master-Datenbank importiert.
3. Es muß ein Abgleich zwischen den Daten in der Master-Datenbank und den Importierten Änderungen der Client-Datenbank erfolgen. Wenn die Client Datenbank Einträge besitzt, die in der Master-Datenbank bereits nicht mehr vorhanden sind, dann müssen diese Daten aus der Client-Datenbank gelöscht werden. Dies geschieht mittels eines SQL-Statements.
4. Alle zulässigen Einträge können dann in die Master-Datenbank übertragen werden.

Wer vielleicht den Eindruck hat, daß Datenbank-Synchronisation mit ausgewachsenen Datenbanken einfacher sei, der irrt gewaltig. Alle die hier getätigten Überlegungen müssen auch bei teuren, kommerziellen Datenbanken angestellt werden.







46.6 Verteilte Datenbanken mit Mail-Synchronisation

Der Datenbankhersteller SYBASE hat mit seiner Datenbank SYBASE Anywhere eine Idee implementiert, die in der Öffentlichkeit wenig bekannt ist. Hier wird ein Abgleich über Mail-Austausch durchgeführt. das Prinzip ist ähnlich dem obigen und verläuft nach folgendem Schema

- Es muß ein zusätzliches Feld vom Typ TIMESTAMP angelegt werden, alle Statements können unverändert übernomemn werden.
- Ein PERL oder PHP3 Skript durchsucht im Abstand von einigen Stunden periodisch die Datenbank auf Veränderungen. Diese Änderungen werden in eine ASCII Datei geschrieben.
- Diese ASCII Datei wird komprimiert und UUENCODED, damit diese als Attachment einfach versendet werden kann.
- Diese Datei wird mit dem Skript **mail -s "Update MySQL" verteiler@intra.net < update.zip.uu** an einen Mail-Verteiler im Intranet gesendet.
- Dieser Verteiler sendet dieses Update an einen bestimmten User in allen Filialen.
- Dort in den Filialen übernimmt der Mail-Server diese Datei und übergibt diese einem Programm, welches die Datei UUDECODED und entkomprimiert in das Verzeichnis der Datenbank ablegt.
- Ein Skript fügt diese Änderungen in die Datenbank ein.

Das Verfahren ist recht einfach. Ein Skript, welches die Datei am Mailerdämon abfängt, UUDECODED und entkomprimiert finden Sie hier: http://www.rent-a-database.de/mysql/decode.txt. Es arbeitet zusammen mit dem SENDMAIL Dämon und funktioniert mit allen UNIX Derivaten zusammen.

Mit diesem Präprozessor, der einen Abgleich von Datenbanken über Mail durchführt, hat man im Prinzip mehrere Probleme gleichzeitig gelöst. Man benötigt keine Standleitung, und man kann zu minimalen Kosten mehrmals täglich einen Abgleich durchführen, indem man einfach ein paar Mails versendet und empfängt. Das Prinzip ist mit allen SQL Datenbanken durchführbar, da es für PERL und PHP3 Interfaces für alle Datenbank- Hersteller gibt. Sogar ein Gemischtbetrieb von mehreren Datenbanken ist möglich.

Bezüglich der Sicherheit braucht man sich ebenfalls keine Gedanken machen, da man Mails über jede Firewall transportieren kann, und man zudem beim Komprimieren mit z.B. ZIP die Datei gleichzeitig verschlüsseln kann.

Wenn man die Daten über Mail-Accounts im Internet austauscht, dann kann man z.B. mit FETCHMAIL unter LINUX oder SOLARIS oder VPOP3 unter Windows NT diese Postfächer regelmäßig leeren, und die Update-Dateien in das Verzeichnis des SQL Servers kopieren. Das bedeutet in der Praxis, daß man alle Filialen nur über eine lokale DIAL-ON-DEMAND Leitung international miteinander vernetzen kann. Standleitungen über Kontinente hinweg, oder quer durch die Bundesrepublik sind somit völlig überflüssig.







46.7 PERL Dämon zur Synchronisation dezentraler Datenbanken

Der Einsatz dieses PERL Dämons erfordert fest IP-Nummern, der Abgleich durchgeführt werden kann. Siehe auch http://www.rent-a-database.de/support/mysql/mysqlsync.txt







46.8 Replikation von MySQL Datenbanken mit PERL in Echtzeit

Hier ein weiteres Script zur **Echtzeit** - Replikation verteilter MySQL - Datenbanken. Das Script stammt von elble@icculus.nsg.nwu.edu und findet sich im Mail-Archiv "http://www.bitmechanic.com/mail-archives/mysql/Jul1999/" name="http://www.bitmechanic.com/mail-archives/mysql/Jul1999/">. Die Kopfinformationen stammen von dem GNU POD-Format und können gelöscht werden. Damit dieses PERL-Script ausgeführt werden kann, müssen PERL und die MYSQL Library für PERL installiert sein: http://www.rent-a-database.de/support/mysql/mysqlrep.txt







46.9 Verschlüsselter Datentransfer

MySQL selber unterstützt keine Datenverschlüsselung. Mit Hilfe des PPTP - Servers unter UNIX/LINUX für Multi-Point-to Point Verbindungen und den eingebauten PPTP Clients bei allen Windows Betriebssystemen ist eine verschlüsselte, sichere Übertragung über unsichere Netzwerke kostenlos verfügbar. IPSec zum Nulltarif ! Die Verschlüsselung betrifft dann das ganze Interface, was bedeutet, daß der gesamte IP-Traffic verschlüsselt wird, also auch die MySQL Datenpakete. Siehe hierzu auch http://www.little-idiot.de/firewall/ Man kann hiermit einen komplett verschlüsselten Server im Internet aufbauen, der nur verschlüsselt mit Clients kommuniziert.







46.10 Verschlüsselte Festplatten

Wer im Internet Datenbanken zentral für den Außendienst lagert, der hat zwei Probleme. Erstens droht die Gefahr von Crackern, die in die Datenbank eindringen, zweitens besteht die Gefahr des Diebstahls des Servers aus den Räumen des Providers. Das erste Problem läßt sich mit Hilfe von IPSec maßnahmen auf allen an das Internet angebundenen Interfaces aus der Welt schaffen. Ohne den korrekten Schlüssel keinen Zugriff auf den Server. Das zweite Problem läßt sich utner LINUX, Windows NT und einigen anderen Derivaten mit TCFS und 3DES-Verschlüsselung lösen. TCFS verschlüsselt eine ganze Festplattenpartition. Nach einem Stromausfall z.B. bootet der Server, danach muß aber per Hand das Passwort für das TCFS eingegeben werden, damit die Partition wieder on-the-fly ver-und entschlüsselt werden kann. Auf dieser lagern dann diebstahlsicher die Unternehmensdaten.







47. SQL Datenbank Beispiel Inventur aus der Zeitschrift i'X

Das beispiel zeigt eine kleine Datenbank in SQL, die aus zwei Tabellen besteht, PERSON und HARDWARE. Alle Abfragen sind in PERL geschrieben. Hier also der Link
http://www.heise.de/ix/artikel/1997/08/150/artikel.html







48. Bild - Datenbank und Bildermanipulation mit ImageMagick

Das schöne an PERL und seinen Modulen ist, daß man hiermit sogar ON-THE-FLY Bilder manipulieren kann (versuchen Sie das mal mit Microsoft Software). Hier eine kurze DEMO mit etwas Code aus dem LINUX-Magazin:

http://www.linux-magazin.de/ausgabe.1999.03/Magick/magick.html







49. Tutorial MySQL Adressen - Datenbank

Dieses Beispiel stammt von Michel Schilli, siehe http://www.perlmeister.com, dem PERLMEISTER im wahrsten Sinne des Wortes .....Darüber hinaus ist es bestens dokumentiert !

Es beschreibt ausführlich den Aufbau einer Adress - Datenbank mit MySQL und PERL. Das besondere daran ist die DBI Schnittstelle, die als MiddleWare eine definierte Schnittstelle zwischen PERL und verschiedensten Datenbanken bietet. Mit nur ein paar Änderungen im Code kann man so zwischen MySQL, ORACLE oder einfach nur einer ASCII Datei als Datenbasis wählen.

Perls generische Datenbankschnittstelle DBI bietet einen Treiber an, der SQL-Abfragen auf Datenbestände ohne eine Datenbank zuläßt - ordinäre Dateien speichern die Tabellendaten in Komma - separierten Einträgen. Ideal für ein kleines CGI-Adreßbuch!




49.1 Datenbank ohne Datenbank

Nicht jeder Provider bietet eine Datenbank an, aber fast alle unterstützen CGI-BIN´s.

Perls generische Datenbankschnittstelle DBI kann man auch mit einem Treiber für ordinäre Dateien ansteuern - statt mit einer Datenbank zu kommunizieren, verwaltet das DBD::File-Modul lesbare Dateien mit Komma - separierten Einträgen, die man mit SQL-Zugriffen abfragen kann !!!

Hinweise zur Installation des Treibers und einiger abhängiger Module, sowie zur Aktivierung des vorgestellten Skripts **addr.pl** finden sich im Abschnitt "Installation" am Ende des Artikels.

Schauen Sie sich hier zu bitte die Abbildung 1 an: http://www.linux-magazin.de/ausgabe.1999.05/DBflat/addr.gif

Abbildung 1 zeigt das Eingangsformular, das das CGI-Skript addr.pl, wenn es einmal in cgi-bin installiert und initialisiert ist, beim ersten Aufruf in den Browser zaubert: Zunächst zeigt addr.pl keinerlei Daten an (auch wenn die Tabelle schon mit Einträgen gefüllt wäre), sondern nur eine Liste der Buchstaben des Alphabets, deren jeder mit einem Link verknüpft ist, der addr.pl nach einem Eintrag suchen läßt, dessen Vorname oder Nachname mit dem betreffenden Buchstaben anfängt. Ein Klick auf den "Alle Einträge" - Link zeigt das ganze Notizbuch an.

Weiter wird ein Suchfeld angezeigt, das Suchstrings entgegennimmt. Falls Suche starten gedrückt wird, sucht addr.pl ebenfalls in den Vorname/Nachname-Feldern der Tabelle nach Übereinstimmungen und zeigt die Ergebnisse in einer Liste an. Klickt der Benutzer auf den Neuer Eintrag-Knopf, wird ein Formular nach Abbildung 2 angezeigt:

name="http://www.linux-magazin.de/ausgabe.1999.05/DBflat/edit.gif>"

Der nach dem Ausfüllen der Felder gedrückte Speichern-Knopf läßt die Daten in die Datenbank wandern. Die Einträge in der Adreßbuchdatei liegen bei dem im Skript verwendeten DBD::File-Treiber in der Datei "addressbook/addressbook" unterhalb des CGI-Verzeichnisses, der Inhalt sieht etwa folgendermaßen aus:

```
id,     fname,  lname,  phone,  email,  addr,   notes
9214031581423,Freddy,Holler,089/1234567,holler@aol.com,"Bon-Scott-Weg 3,89834
Zuffenhausen",
"Alte Email: fredy@aol.com"
9214038931493,Herbert,Rigatoni,08234/234435,
herbertR@yahoo.com,"In der Grube 24, 82342 Kutzenbach","Geburtstag: 1.3."
```

Die erste Zeile legt hierbei die Spaltennamen fest. Sie lauten genauso wie die später verwendeten CGI-Parameternamen, das erspart Kopfschmerzen bei der Programmierung. "fname" steht dabei für den Vornamen (First Name), lname für den Nachnamen (Last Name) usw. Einträge in den Datenzeilen werden durch Kommata getrennt. Einträge, die Leerzeichen enthalten, werden in doppelte Anführungszeichen eingeschlossen und eventuelle Sonderzeichen (doppelte Anführungszeichen und Kommata) entsprechend maskiert. Die erste Spalte (id) jeder Zeile weist dem Eintrag eine eindeutige ID zu. Sie setzt sich zusammen aus der gegenwärtigen Uhrzeit (Rückgabewert des time()-Kommandos) und der PID des aktuellen CGI-Prozesses beim Anlegen des Eintrags.

49.2 Suchen und Finden

Eine gefüllte Datenbank liefert Ergebnisse auf Suchabfragen in Tabellenform nach Abbildung 3. Das Namensfeld jedes Eintrags ist mit einem Link unterlegt, klickt man darauf, springt der Browser auf die Editierseite und füllt die Felder dort gleich mit den Daten des ausgewählten Eintrags. Der Speichern-Knopf aktualisiert den Datenbank-Eintrag entsprechend den Formularfeldern, ein Druck auf den Delete-Knopf löscht den Eintrag:

name="http://www.linux-magazin.de/ausgabe.1999.05/DBflat/list.gif>"

Was addr.pl im einzelnen tut, ob es ein Eingabeformular darstellt oder einen neuen Eintrag anlegt oder das Ergebnis einer Suchanfrage anzeigt, bestimmen die CGI-Parameter mit denen es aufgerufen wird. Folgende Szenarien steuert addr.pl:

- Datenbanktabelle initialisieren

  addr.pl init=1

  Dieser Aufruf erzeugt die Datenbank-Tabelle mit einem CREATE-Kommando aus dem SQL-Fundus.
- Suchabfrage mit Ergebnisanzeige

  addr.pl search=A

  Die Such-Abfrage fördert Einträge hervor, deren Vorname oder Nachname mit A angehen und zeigt die Ergebnisse in einer Tabelle an. Wird der search-Parameter leergelassen (aber dennoch definiert mit search=), zeigt addr.pl eine vollständige Liste aller bestehenden Einträge in einer Tabelle an.
- Formular zur Eingabe eines neuen Eintrags anzeigen

  addr.pl edit=1
- Formular zur Aktualisierung eines bestehenden Eintrags anzeigen

  addr.pl edit=1 id=9214031581423

  Jeder Tabelleneintrag enthält eine eindeutige ID, so daß addr.pl einmal gefundene Zeilen beim nächsten Aufruf schnell identifizieren und Manipulationen vornehmen kann (update, delete).
- Neuen Eintrag aus den ausgefüllten Formularfeldern generieren

  addr.pl insert=<gesetzt> fname=... lname=...

  Eintrag mit den ausgefüllten Formularfeldern aktualisieren

  addr.pl insert=<gesetzt> id=9214031581423 fname=... lname=...

  Eintrag löschen

  addr.pl delete=<gesetzt> id=9214031581423

Das Skript addr.pl im Detail:

Listing addr.pl zeigt die Implementierung des Web-Adreßbuches. Zeile 7 holt das CGI-Modul, die angegebenen Tags lassen es die Standard-HTML- und die Tabellen-Funktionen exportieren. Das CGI::Carp-Modul sorgt dafür, daß der Browser bei auftretenden Fehlern nicht das blöde Internal Server Error anzeigt, sondern einen aufschlußreiche Fehlermeldung. Die Zeilen 11 bis 14 spezifizieren die Parameter für den DBI-Flatfile-Treiber. $DB_DIR gibt das Verzeichnis unterhalb des cgi-bin-Verzeichnisses an, das die Tabellendaten als Datei enthält.

Zeile 20 nimmt die Verbindung mit der "virtuellen" Datenbank auf, die Zeilen 23 und 24 geben den CGI-Header und die Überschrift aus, die in jedem Fall im Dokument steht und färben den Hintergrund der Seite weiß ein. Dann scheiden sich die Wege: Der if-Block ab Zeile 26 wird angesprungen, falls ein Benutzer die Formularfelder für einen neuen Eintrag ausgefüllt und den Speichern-Knopf gedrückt hat. Der map-Befehl in Zeile 28 übergibt der insert_rec-Funktion, die die eigentliche Datenbank-Aktualisierung vornimmt, die Formulardaten, indem er für alle Elemente in @dbfields die param-Funktion des CGI-Moduls aufruft und so die entsprechenden CGI-Parameter entgegennimmt und weiterreicht.

insert_rec selbst steht ab Zeile 158 in addr.pl, nimmt das hereingereichete DB-Handle und die Formularparameter entgegen und setzt den SQL-Insert/Update-Befehl an die Datenbank ab. Ist der CGI-Parameter id gesetzt, handelt es sich um eine Aktualisierung eines bestehenden Records und Zeile 167 definiert einen SQL-Update-Befehl. Hier wie auch an anderen Stellen leistet der qq-Operator, der mehrzeilige Strings mit doppelten Anführungszeichen umschließt, nützliche Dienste. Fehlt andererseits id, handelt es sich um einen neuen Eintrag und Zeile 180 kreiert einen SQL-Insert-Befehl.

Zurück zur Hauptschleife: Die page_header-Funktion, die in Zeile 29 aufgerufen wird und ab Zeile 99 implementiert ist, klatscht das kleine Link-Alphabet, das in den Abbildungen 1 und 3 jeweils oben im Fenster zu sehen ist, dorthin und schreibt auch noch das Such-Feld samt den zwei Buttons auf die Seite. Die url()-Funktion aus dem CGI-Modul liefert hierzu den URL des gegenwärtig laufenden Skripts.

Ab Zeile 31 steht der Code zum Löschen eines Eintrags. Jede Zeile in der Datenbanktabelle enthält neben den Adreßbuchdaten auch noch eine eindeutige ID, die als verstecktes (hidden) Feld auf der Seite steht, die die Formularfelder zum Aktualisieren eines Eintrags darstellt. Drückt der Benutzer auf den Knopf Eintrag löschen, sendet der Browser neben den aktualisierten Feldern auch noch die ID mit und addr.pl kann einen DELETE-Befehl losschicken, der mit seinem Tintenkiller genau über die richtige Zeile der Tabelle fährt. Drückte der Benutzer entweder auf den Knopf Neuer Eintrag oder aber auf einen erleuchteten Namen der der Ergebnisliste, wird der Codeblock ab Zeile 40 angesprungen, da der Parameter edit in diesen Fällen gesetzt ist.

Diese zwei Fälle unterscheiden sich dahingehend, daß ein angeklickter Eintrag der Ergebnisliste den id-Parameter setzt. In diesem Fall muß addr.pl vor dem Darstellen der Felder die Werte aus der Datenbank übernehmen. Hierzu erzeugt es in Zeile 44 einen SQL-Select-Befehl, der die Daten aus der Datenbank holt. Die Zeilen 51 bis 56 holen das Ergebnis des Queries ab, wegen der eindeutigen ID im SELECT ist das Ergebnis stets eine einzelne Zeile. Die fetch-Methode im while-Kopf liefert eine Referenz auf einen Array zurück, dessen Elemente die Spaltenwerte des Tabelleneintrags beinhalten. Da die Tabellenzeile zusätzlich zu den in @dbfields aufgelisteten Spalten als erstes Element die id-Spalte führt, startet der Index $i in Zeile 52 mit dem Wert 1 statt des sonst üblichen 0 .

Der Aufruf der param-Methode in Zeile 54 manipuliert die CGI-Eingangsparameter und gaukelt den

nachfolgenden Abfragen vor, der Benutzer hätte die Adreßdaten des selektierten Eintrags selbst eingetragen - derweil stammen sie aus der Datenbank. Die Zeilen 59 bis 80 geben eine zweispaltige HTML-Tabelle aus, die das Formular zum Anlegen/Editieren eines Adreßeintrags nach Abbildung 2 in den Browser zeichnet.

Für den Fall, daß der Benutzer eine Suchanfrage startete oder einen Buchstaben im Reiter-Alphabet des Seitenkopfes anklickte, ist der CGI-Parameter search gesetzt, entsprechend springt addr.pl den Block ab Zeile 82 an. Für den Buchstabenklick enthält search den entsprechenden Buchstaben, wurde etwas ins Suchfeld eingetragen und der Suche starten-Knopf gedrückt, steht in search der Suchbegriff. Die Funktion keyword_search übernimmt in beiden Fällen die Suche, sie ist ab Zeile 118 definiert. Dort holt eine SQL-Abfrage passende Records aus der Datenbank, indem sie mittels des CLIKE-Konstrukts überprüft, ob Vor- oder Nachname eines Eintrags mit dem gegebenen Suchausdruck beginnen, Groß- und Kleinschreibung werden ignoriert. Für einen leeren Suchstring liefert keyword_search großzügigerweise einfach alle Einträge der Tabelle.

Die while-Schleife ab Zeile 145 gibt die Treffer in einer HTML-Tabelle aus, indem sie Vor- und Nachnamen zu einer Tabellenspalte zusammenfaßt und einen HTML-Link drumherum baut, der die CGI-Parameter edit auf 1 und id auf die in der Datenbank gefundene ID des Eintrags setzt, so daß das Skript bei einem Klick auf den Eintrag sofort den Eintrag in der Datenbank referenzieren kann. Der Block ab Zeile 86 kommt nur bei der Installation des Skripts kurz zum Einsatz und ruft die Initialisierungsfunktion init_db auf, die ab Zeile 190 definiert ist, und das Unterverzeichnis der Pseudo-Datenbank erzeugt. Weiter setzt sie einen SQL-Create-Befehl ab, der die Pseudo-Tabelle anlegt. Ist überhaupt kein CGI-Parameter gesetzt (wie beim ersten Aufruf des Skripts), kommt Zeile 90 zum Einsatz und zeichnet lediglich den Seitenkopf mit dem Suchfeld und dem Reiteralphabet.

```
#!/usr/bin/perl -w
##################################################
# CGI Address Book
# 1999, mschilli@perlmeister.com
##################################################

use CGI qw/:standard :html3/;
use CGI::Carp qw/fatalsToBrowser/;
use DBI;

my $DB_DIR      = "./addressbook";
my $DB_DSN      = "DBI:CSV:f_dir=$DB_DIR";
my $DB_USER     = "";
my $DB_PASSWD   = "";

my @dbfields = qw/fname lname phone email addr
                  notes/;
my $dbflist  = join(', ', @dbfields);

my $dbh = DBI->connect($DB_DSN, $DB_USER,
   $DB_PASSWD) or die "Cannot connect to DB";

print header(), start_html(-BGCOLOR => 'white'),
      h1("Adreßbuch");
```

```
if(param('insert')) {
    # Insert/Update record according to form fields
    insert_rec($dbh, map { $dbh->quote(param($_)) } @dbfields);
    page_header();

} elsif(param('delete')) {
    # Delete a record according to ID field
    my $id = $dbh->quote(param('id'));
    $dbh->do(<<EOT) or die "Cannot delete data";
        DELETE FROM addressbook
        WHERE id = $id
EOT
    page_header();

} elsif(param('edit')) {
    # Display fields for inserting/updating a rec
    if(param('id')) {
        $id = $dbh->quote(param('id'));
        # ID exists - Get record and preset fields
        my $sql = qq[SELECT id, $dbflist
                     FROM addressbook
                     WHERE id = $id];
        my $cursor = $dbh->prepare($sql) or
                     die "Cannot select ($sql)";
        $cursor->execute() or die "SQL failed";

        while(defined($row = $cursor->fetch)) {
            my $i = 1;
            foreach $field (@dbfields) {
                param($field, $row->[$i++]);
            }
        }
    }

    print start_form(),
      hidden(-name => 'id'),
      table({"border" => 1},
      TR(td("Vorname:"),
         td(textfield(-name => 'fname'))),
      TR(td("Nachname:"),
         td(textfield(-name => 'lname'))),
      TR(td("Telefon:"),
         td(textfield(-name => 'phone'))),
      TR(td("Email:"),
         td(textfield(-name => 'email'))),
      TR(td("Adresse:"),
         td(textarea(-name => 'addr', -rows => 3))),
      TR(td("Notizen:"),
```

```
          td(textarea(-name => 'notes', -rows => 3))),
      );

    print submit(-name  => 'insert',
                 -value => 'Speichern'),
          submit(-name  => 'delete',
                 -value => 'Eintrag löschen'),
          end_form();

} elsif(defined param('search')) {
    page_header();
    keyword_search($dbh, param('search'));

} elsif(param('init')) {
    page_header();
    init_db($dbh);

} else {
    page_header();
}

print end_html();

$dbh->disconnect();  # Datenbankverbindung lösen.

#####################################################
sub page_header {
#####################################################
    print start_form();
    foreach $letter ('A'..'Z') {
        print a({href => url() .
          "?search=$letter"}, "$letter ");
    }
    print a({href => url() .  "?search="},
            " Alle Einträge"),
          p("Suchbegriff:",
            textfield(-name => 'search'),
            submit(-name => 'Search',
                   -value => 'Suche starten'),
            submit(-name => 'edit',
                   -value => 'Neuer Eintrag'));
    print end_form();
}

#####################################################
sub keyword_search {
#####################################################
    my ($dbh, $keyword) = @_;
    my $cursor;
```

```
    my $where_clause = "";

    if($keyword ne "") {
        $keyword = $dbh->quote("$keyword%");
        $where_clause = qq[
            WHERE fname CLIKE $keyword OR
                  lname CLIKE $keyword];
    }

    my $sql = qq[ SELECT id, $dbflist
                  FROM addressbook
                  $where_clause
                  ORDER BY lname];

    $cursor = $dbh->prepare($sql) or
              die "Select failed: $sql";

    $cursor->execute() or
        die "Can't execute ($sql): ", $cursor->errstr;

    print "<TABLE BORDER=1>\n";
    print TR(map { th($_) }
      qw/Name Telefon Email Adresse Notizen/);

    while(defined(my $row = $cursor->fetch)) {
        print TR(td(
        a({href => url() . "?id=$row->[0]=1"},
          "$row->[2], $row->[1]"),
        td("$row->[3]"),
        td("$row->[4]"), td("$row->[5]"),
        td("$row->[6]"),
        )), "\n";
    }
    print "\n";
}

#####################################################
sub insert_rec {
#####################################################
    my($dbh, $fname, $lname, $phone,
       $email, $addr, $notes) = @_;

    if(param('id')) {
        # ID there, it's an update!
        my $id = $dbh->quote(param('id'));

        my $sql = qq[
          UPDATE addressbook
          SET id=$id, fname=$fname,
```

```
                lname=$lname, phone=$phone,
                email=$email, notes=$notes
            WHERE id = $id];

        $dbh->do($sql) or die "Update failed ($sql)";

    } else {
        # ID not there, it's a new record!
        my $id = time . $$;  # Generate ID
        $id = $dbh->quote($id);

        my $sql = qq[
          INSERT INTO addressbook
          (id, $dbflist)
          VALUES
                  $email, $addr, $notes)];
        $dbh->do($sql) or die "Insert failed ($sql)";
    }
}

####################################################
sub init_db {
####################################################
    my $dbh = shift;

    if(! -d $DB_DIR) {
        mkdir($DB_DIR, 0755) ||
            die "Cannot create dir $DB_DIR";
    }

    $dbh->do(<<'EOT') or die "Cannot create table";
        CREATE TABLE addressbook (
            id     char(20),
            fname  char(40),  lname char(40),
            phone  char(20),  email char(40),
            addr   char(100), notes char(100)
        )
EOT
}
```






50. Projekt MySQL und PHP3 für Fortgeschrittene

Dieses Projekt ist speziell für diejenigen gedacht, die mit PHP3 Verbindungen zu anderen Servern im Internet aufbauen möchten. Beispielsweise wird hier detailliert die Abfrage der DENIC Datenbank beschrieben, in der alle Besitzer von Domains mit Name, Adresse, Anschrift, Faxnummer, u.s.w. enthalten sind. In Verbindung mit MySQL lassen sich natürlich so ganz andere Marketingmöglichkeiten realisieren.....

```
<html>
<body>
<?
    echo("Hello World\n");
?>
```

Nunja ich denke, daß ein "Hallo Welt !" niemanden mehr hinterm Ofen hervorlocken kann, es ist trotzdem immer ein guter Test, um festzustellen, ob PHP3 implementiert ist ....

```
shell> mysql -uUSERNAME -pPASSWORD

mysql> create database phptest;
Query OK, 1 row affected (0.13 sec)

mysql> create table TEST
    -> (
    -> ID        int auto_increment primary key,
    -> Name      varchar(32),
    -> Age       int,
    -> Salary    int
    -> );
Query OK, 0 rows affected (0.11 sec)


mysql> insert into TEST values (0,'Billy Bob Barker','87','93000');
Query OK, 1 row affected (0.09 sec)

mysql> insert into TEST values (0,'Sniffling Sam Sloth','23','12000');
Query OK, 1 row affected (0.01 sec)

mysql> insert into TEST values (0,'Roy Bartley','31','87000');
Query OK, 1 row affected (0.01 sec)

mysql> insert into TEST values (0,'Leroy Longrad','45','63000');
Query OK, 1 row affected (0.01 sec)

mysql> insert into TEST values (0,'Amy Antwerp','37','34000');
Query OK, 1 row affected (0.01 sec)

mysql> insert into TEST values (0,'Kim Kruger','57','76000');
Query OK, 1 row affected (0.01 sec)


mysql> select * from TEST;
```

```
+----+--------------------+------+--------+
| ID | Name               | Age  | Salary |
+----+--------------------+------+--------+
|  1 | Billy Bob Barker   |   87 |  93000 |
|  2 | Sniffling Sam Sloth |   23 |  12000 |
|  3 | Roy Bartley        |   31 |  87000 |
|  4 | Leroy Longrad      |   45 |  63000 |
|  5 | Amy Antwerp        |   37 |  34000 |
|  6 | Kim Kruger         |   57 |  76000 |
+----+--------------------+------+--------+
6 rows in set (0.16 sec)
```

Create a new file, "testdb.php3" and type in the following:

```
<html>
<body>
<?
    require("util.php3");
    $sql = new MySQL_class;
    $sql->Create("phptest");

    echo("Database connection successful.\n");
?>
```

Das sind die Routinen in util.php3

```
<?
/*
 * Utility routines for MySQL.
 */

class MySQL_class {
    var $db, $id, $result, $rows, $data, $a_rows;
    var $user, $pass, $host;

    /* Make sure you change the USERNAME and PASSWORD to your name and
     * password for the DB
     */

    function Setup ($user, $pass) {
        $this->user = $user;
        $this->pass = $pass;
    }

    function Create ($db) {
        if (!$this->user) {
            $this->user = "USERNAME";
        }
        if (!$this->pass) {
            $this->pass = "PASSWORD";
        }
        $this->db = $db;
        $this->id = @mysql_pconnect($this->host, $this->user, $this->pass)
```

```
or
            MySQL_ErrorMsg("Unable to connect to MySQL server: $this->host :
'$SERVER_NAME'");
        $this->selectdb($db);
    }

    function SelectDB ($db) {
        @mysql_select_db($db, $this->id) or
            MySQL_ErrorMsg ("Unable to select database: $db");
    }

    # Use this function is the query will return multiple rows.  Use the
Fetch
    # routine to loop through those rows.
    function Query ($query) {
        $this->result = @mysql_query($query, $this->id) or
            MySQL_ErrorMsg ("Unable to perform query: $query");
        $this->rows = @mysql_num_rows($this->result);
        $this->a_rows = @mysql_affected_rows($this->result);
    }

    # Use this function if the query will only return a
    # single data element.
    function QueryItem ($query) {
        $this->result = @mysql_query($query, $this->id) or
            MySQL_ErrorMsg ("Unable to perform query: $query");
        $this->rows = @mysql_num_rows($this->result);
        $this->a_rows = @mysql_affected_rows($this->result);
        $this->data = @mysql_fetch_array($this->result) or
            MySQL_ErrorMsg ("Unable to fetch data from query: $query");
        return($this->data[0]);
    }

    # This function is useful if the query will only return a
    # single row.
    function QueryRow ($query) {
        $this->result = @mysql_query($query, $this->id) or
            MySQL_ErrorMsg ("Unable to perform query: $query");
        $this->rows = @mysql_num_rows($this->result);
        $this->a_rows = @mysql_affected_rows($this->result);
        $this->data = @mysql_fetch_array($this->result) or
            MySQL_ErrorMsg ("Unable to fetch data from query: $query");
        return($this->data);
    }

    function Fetch ($row) {
        @mysql_data_seek($this->result, $row) or
            MySQL_ErrorMsg ("Unable to seek data row: $row");
        $this->data = @mysql_fetch_array($this->result) or
            MySQL_ErrorMsg ("Unable to fetch row: $row");
    }

    function Insert ($query) {
        $this->result = @mysql_query($query, $this->id) or
            MySQL_ErrorMsg ("Unable to perform insert: $query");
        $this->a_rows = @mysql_affected_rows($this->result);
    }
```

```
    function Update ($query) {
        $this->result = @mysql_query($query, $this->id) or
            MySQL_ErrorMsg ("Unable to perform update: $query");
        $this->a_rows = @mysql_affected_rows($this->result);
    }

    function Delete ($query) {
        $this->result = @mysql_query($query, $this->id) or
            MySQL_ErrorMsg ("Unable to perform Delete: $query");
        $this->a_rows = @mysql_affected_rows($this->result);
    }
}

/* ********************************************************************
 * MySQL_ErrorMsg
 *
 * Print out an MySQL error message
 *
 */

function MySQL_ErrorMsg ($msg) {
    # Close out a bunch of HTML constructs which might prevent
    # the HTML page from displaying the error text.
    echo("\n");
    echo("\n");

    # Display the error message
    $text  = "<font color=\"#ff0000\" size=+2><p>Error: $msg :";
    $text .= mysql_error();
    $text .= "\n";
    die($text);
}
?>
```






51. MySQL Performance Benchmarks

Die Performance von MySQL hängt stark von der Art des Interfaces ab. Legt man den offiziellen TPC-Benchmark zugrunde, siehe http://www.tpc.org, dann ist MySQL mit dem MM JDBC-Klasse 4 Treiber mit kontinuierlichen 100 Transaktionen/Sekunde bei 100 Clients nicht zu schlagen. Auch bei 300 simultanen Clients, einer Zahl, die im Internet durchaus häufiger auftreten kann, liefert MM-JDBC Treiber immer noch 60 Transaktionen/Sekunde. Der TWZ JDBC Klasse 4 Treiber sinkt bei 100 Clients auf 40 Transaktionen/Sekunde ab. Diese Werte sind im übrigen schon auf einem 600 DM Pentium Rechner ohne Probleme zu überbieten....(mein Testserver), also kann man sich einen DUAL CELERON und U2W SCSI Festplatten sparen.....JDBC-ODBC Treiber schaffen ca. 10 Transaktionen/Sekunde mit bis zu 10 Clients gleichzeitig. Das ist wohl die Geschwindigkeit, die man unter ACCESS gewohnt ist....;-) Die Benchmarks und deren Ergebnisse findet man im übrigen auf http://www.worldserver.com/mm.mysql/index.html#downloads







52. Alternative SQL Datenbanken

Neben MySQL gibt es natürlich auch noch Alternativen, die hier auch nicht vorenthalten werden sollen. Dies können manchmal mehr, manchmal weniger, als MySQL. In jedem Fall sollte man sich auch diese einmal anschauen.







52.2 HyperSQL für Einsteiger

Kapitel in Arbeit !







52.3 Streamstore Datenbank

http://www.bluestream.com/products/streamstore/index.htm ist eine kleine Datenbank in JAVA geschrieben. Sie unterstützt als reine JAVA - Anwendung Index und Table-Cursors und ermöglicht somit das Blättern durch die Datensätze. Sie unterstützt Transaktionen mit Rollback und Commit.







52.4 Instantdb

http://www.instantdb.co.uk/ Ist auch eine recht preiswerte, sehr schnelle Datenbank mit JDBC Klasse 4 Interface, die vollständig in JAVA geschrieben ist.







53. MySQL unter Microsoft Windows 98/NT

Die Installation von MySQL unter Windows 98 der NT ist recht einfach, daher ist eine ausführliche Beschreibung, wie unter UNIX nicht notwendig. Unter Windows 95 ist MySQL ebenfalls lauffähig, jedoch leidet Windows 95 an einem Memory Leck, sodaß bei jeder Abfrage ca. 200 Byte an RAM verlorengehen. MySQL läßt sich unter Windows NT als Dienst starten und läuft auch dort sehr zuverlässig. MySQL kann auch parallel zu anderen Datenbanken betrieben werden.







53.1 Installation von MySQL unter Windows 98/NT

Hierzu lädt man sich die **SHAREWARE** Version von MySQL von dem Server http://www.tcx.se oder besser von einem der angegebenen Mirror.

Die hier verwendete ZIP Datei heißt **mysql-shareware-win32-1.1.zip** Wohlgemerkt handelt es sich hierbei um Shareware, die 170 Euro für die **unlimited** Version kostet. Angesichts der gegenüber UNIX äußerst einfachen und angenehmen Version sollte man diese 170 Euro überweisen. Die Entwickler sind äußerst kulant und gewähren über die Mailing-Listen auch kostenlosen Support. Falls Sie kommerziellen Support in Anspruch nehmen möchten, so finden Sie zu Beginn des Handbuches wichtige Hinweise.

Mit Hilfe des WINZIP, zu finden auf http://www.tucows.com können Sie dies Datei entpacken. Legen Sie hierzu ein Verzeichnis **c:\mysql** an und extrahieren Sie alle Dateien dort hinein. Dort befindet sich dann eine Datei **setup.exe**. Mit einem Klick auf die Datei wird dann **MySQL** installiert.

Leider ist die Dokumentation völlig auf UNIX ausgerichtet, sodaß es für einen Einsteiger recht schwer ist, die Zusammenhänge der einzelnen Programme zu erkennen.

Nun müssen noch einige Anpassungen vorgenommen werden. Die erste Anpassung betrifft die Datei **c:\windows\hosts.sam**. Benennen Sie diese in **hosts** um, und tragen Sie folgende Zeilen noch ein:

```
127.0.0.1       localhost
127.0.0.1       win98
10.0.0.1        win98
```

Anstelle von 10.0.0.1 müssen Sie die TCP/IP Nummer Ihrer Arbeitsstation eintragen. Falls Sie diese nicht kennen, so tippen Sie in der DOS-SHELL einfach: **ipconfig** ein. Die Ausgabe sagt Ihnen, welche IP-Nummer Ihr Host im Moment hat. In einigen Netzwerken mit NT-Servern werden IP-Nummern aus einem Pool vergeben. Das nennt man DHCP und ist völliger Mist, da Sie nun immer eine andere IP-Nummer nach jedem Neustart der Arbeitsstation erhalten. Wenden Sie sich hierzu an Ihren Systemadministrator und bitten ihn, für Ihre Arbeitsstation eine feste IP-Nummer im DHCP Server zu reservieren. Das ist ca. 1 Minute Arbeit (Wenn man sich auskennt). Alternativ vergeben Sie für Ihre Netzwerkkarte eine zusätzliche IP-Nummer, die dann fest eingetragen ist. Diese sollten Sie unbedingt aber mit Ihrem Systemadministrator vorher absprechen, da eventuell ansonsten ganze Server außer Gefecht gesetzt sind, oder Router und Firewalls verrückt spielen.




53.2 Die MySQL Binaries - kurz erklärt !

Im Unterverzeichnis **bin/** des Installationsverzeichnisses **c:\mysql** finden sich alle wichtigen Binaries: Einige Beispiele erfordern, daß der **mysqld** schon gestartet ist. Klicken Sie einfach mit der Maus auf die Datei **mysqld.exe** im Verzeichnis **c:\mysql\bin**. Der Serverdämon ist nun gestartet. Nun zu den Binaries:

- **mysqld.exe** ist der MySQL Serverdämon. Startet man mysqld.exe ohne weitere Optionen, so startet dieser unter NT automatisch als Dienst / Service. Falls Sie dieses zuerst nicht möchten, so sollten Sie **mysqld --standalone** angeben. Zum Beenden von mysql sollten Sie immer den Befehl: **mysqladmin -u root shutdown** verwenden, damit der mysql Server auch ordnungsgemäß heruntergefahren wird. Datenverluste sind ansonsten unvermeidlich. Um MySQL als Dienst / Service zu installieren, sollten Sie **mysqld --install** verwenden, um diesen Dienst wieder zu entfernen, geben Sie einfach das Kommando **mysqld --remove** an. Es ist auch völlig egal, ob der Serverdämon als Administrator oder mit einem normalen User-Account gestartet wird.
- **isamchk.exe** ist das Programm, mit welchem zerstörte Datenbanken restauriert oder nachlängerem Betrieb die Tabellen von Müll bereinigt werden können.
- **mysql.exe** ist der MySQL Client, mit welchem DML (Data manipulation Language) Kommandos an die Datenbank gegeben werden. Das sind z.B. das SELECT Kommando oder noch einfacher: **show variables;**. Die Ausgabe:

```
+--------------------+--------------------------+
| Variable_name      | Value                    |
+--------------------+--------------------------+
| back_log           | 5                        |
| basedir            | c:\mysql\                |
| datadir            | .                        |
| join_buffer        | 131072                   |
| key_buffer         | 1048540                  |
| language           | c:\mysql\share\english\  |
| max_allowed_packet | 65536                    |
| max_connections    | 90                       |
| max_join_size      | 4294967295               |
| max_sort_length    | 1024                     |
| net_buffer_length  | 8192                     |
| port               | 3306                     |
| record_buffer      | 131072                   |
| skip_locking       | 0                        |
| socket             | /tmp/mysql.sock          |
| sort_buffer        | 2097116                  |
| table_cache        | 64                       |
| thread_stack       | 65536                    |
| tmp_table_size     | 1048576                  |
+--------------------+--------------------------+
19 rows in set (0.00 sec)

mysql> show variables;
```

Bitte lesen Sie hierzu die entsprechenden Abschnitte über dieses Kommando unter UNIX und vergleichen Sie. MySQL ist ursprünglich völlig unter UNIX entwickelt worden. Logischerweise hat man die Werkzeuge nun auf DOS einfach portiert. Die Syntax ist also weitestgehend dieselbe, nur müssen Sie stets alle

Kommandos in der DOS-Shell absetzen. Falls Sie also zu der Klick Generation gehören, die alles gerne mit Benutzeroberfläche bedienen möchten, so kann auch diesem Wunsch entsprochen werden. Mehr dazu später.

- **mysqladmin.exe** ist das Toolkit, mit welchem man Datenbanken anlegen, löschen, sperren, exportieren u.s.w. kann. Dieses Toolkit kümmert sich um die Kommandos der Data Definition Language (DDL) von SQL:

```
Where command is a one or more of: (Commands may be shortened)
  create databasename   Create a new database
  drop databasename     Delete a database and all its tables
  flush-hosts           Flush all cached hosts
  flush-logs            Flush all logs
  flush-tables          Flush all tables
  kill id,id,...        Kill mysql threads
  password newpassword  Change old password to newpassword
  processlist           Show list of active threads in server
  reload                Reload grant tables
  refresh               Flush all tables and close and open logfiles
  shutdown              Take server down
  status                Gives a short status message from the server
  variables             Prints variables available
  version               Get version info from server

C:\mysql\bin>mysqladmin create test
C:\MYSQL\BIN\MYSQLA~1.EXE: create of 'test' failed
error: 'Can't create database 'test'. Database exists'

C:\mysql\bin>mysqladmin create stepken
Database "stepken" created.

C:\mysql\bin>
```

Wie man hier sehen kann, ist die Datenbank test schon angelegt. Alles weitere ist analog zu der Anleitung der UNIX Version. Zum Beenden von mysql sollten Sie immer den Befehl: **mysqladmin -u root shutdown** verwenden, damit der mysql Server auch ordnungsgemäß heruntergefahren wird. Datenverluste sind ansonsten unvermeidlich.

- **mysqlshow** ohne Parameter aufgerufen zeigt alle verfügbaren Datenbanken an. Der Befehl **mysqlshow test** zeigt alle Tabellen in der Datenbank test an. **mysqlshow --?** zeigt eine Menge weiterer Parameter (wie übrigens alle anderen DOS-Kommandos auch), deren Bedeutung im abschnitt über MySQL unter UNIX klar erschließt.
- **mysqlimport** ist das Geheimnis der Datenübernahme von anderen Datenbank in MySQL aus Files. Mehr dazu in dem UNIX Abschnitt
- **replace** Dient dazu, Datenbanken zur Laufzeit auszutausche

---






53.3 Windows und UNIX/NT kombiniert

Es ist fast immer gewünscht, die Daten zentral auf einem Server abzulegen, und von einem Windows Arbeitsplatz diese MySQL Datenbank zu administrieren. Dies ist selbstverständlich ebenfalls möglich. Hierzu muß auf dem Server die MySQL Datenbank installiert sein. Dies kann UNIX (LINUX) oder ein NT-Server sein. Mit Hilfe des Clients der Microsoft Version von MySQL kann man dann einfach über das Netzwerk auf den Server zugreifen, und die SQL Datenbank komplett administrieren. Das funktioniert auch über ISDN und das Internet. Hierzu muß man beim Start des MySQL Client für Windows einfach einen anderen Host eintragen. Der Client kommuniziert dann über das Netzwerk mit dem MySQL Server.







53.4 MySqlManager

Der MySqlManager sticht beim Anschauen des Verzeichnisses **c:\mysql\bin** durch sein eigenes Icon hervor. Nachdem man den **mysqld.exe** Serverdämon gestartet hat, kann man nur mit dem **MySqlManager** nun beginnen, die Datenbank zu Administrieren.

Starten Sie ihn ! Es öffnet sich das MySqlManger-Fenster mit einem weiteren Fenster darin. Klicken Sie nun auf **Test** und **MySQL**. Es rollen sich weitere Verzeichnisse aus. Klicken Sie nun auf **mysql**. Es zeigen sich die Verzeichnisse: **user, host, db**. Klicken Sie auf **user**. Was Sie nun sehen, sind die User-Rechte (Siehe auch GRANT) der Tabelle **mysql** in der Datenbank **Test**. Klicken Sie auf das [-] neben **User**, damit sich die Rechte-Tabelle (grant table) wieder einrollt. Klicken Sie auf **db**. Hier sehen Sie nochmals die Grant-Tabellen für die Tabelle **db** mit deren Struktur, also den Spalten und deren Datenformat. Diese Datenbanken und Tabellen sind in MySQL schon vorbereitet, als DEMO zum ansehen. Markieren Sie nun z.B. das oberste Verzeichnis **MySQL** mit der Maus. Gehen Sie nun auf das Menü **Tools** und **SQL Query**. Es öffnet sich ein Fenster. Klicken Sie das Fenster **Query** aktiv. Tippen Sie folgendes ein: **select * from user** und klicken Sie auf den grünen Pfeil. Das Fester **Results**, in welchem Ausgaben und Fehlermeldungen angezeigt werden, meldet: **No Database Selected**. Diese Ausgabe ist korrekt, da wir dem Datenbankserver noch nicht gesagt haben, welche Datenbank wir eigendlich abfragen möchten. Tippen Sie daher im Feld **Query** folgendes ein: **use mysql**. Damit sagen wird dem Datenbank-Server, daß er nun für uns (unsere MySQLManager-Session) die Datenbank **mysql** zu wählen hat. Wiederholen Sie nun den Befehl **select * from mysql**. Sie sehen nun im **Result** Fenster folgende Ausgabe:

```
Host            User    Password        Select_priv     Insert_priv .....
---             ---     -----           ------          --------
localhost       root                    Y               Y
%
localhost                               Y               Y       ....
%               root                    Y               Y       .....

(4 row(s) affected)
```

Die Ausgabe sagt uns, daß der **MySqlAdmin** Manager uns als **root** von der lokalen Maschine (wird haben ja mysqld und den Manager auf derselben Arbeitsstation gestartet) aus angemeldet hat. Ein Passwort ist nicht erforderlich gewesen, was schnellstens geändert werden muß.

Sie können sich mit diesem Manager natürlich auch an andere Datenbanken anbinden, und dort versuchen, Abfragen zu starten. Hierzu finden Sie im Menü **Tools** ein Menüpunkt **Register Server**, mit welchem Sie sich einfach an andere Datenbanken anbinden können. Hier werden Sie eventuell nach Username und Passwort gefragt, bevor Ihre Exkursion beginnen kann.

Der MySqlManager kann noch etwas mehr, allerdings gehört dieses Programm nicht unbedingt zu den komfortabelsten Managern, die es gibt. Wir möchten Ihnen nun einige Manager vorstellen, die nun wirklich zu den feinsten gehören, die die Welt der freien Software zu bieten hat:







54. MySQLWinAdmin

Sie finden den Datenbankmanager in unserem Supportverzeichnis auf dem Server http://www.rent-a-database.de.

Legen Sie hierzu ein Verzeichnis **c:\MySQLWinAdmin** an, in welches Sie mit dem WINZIP Programm das EXE File und die DLL hineinkopieren. Starten Sie nun den **MySQLWinAdmin**. Es erscheint ein Fenster, in welchem offensichtlich noch kein Server angebunden ist. Klicken Sie nun auf den einzigen aktiven Menüpunkt. Es öffnet sich ein Fenster. In der ersten Zeile können Sie die Verbindung mit einem Namen Ihrer Wahl versehen, z.B. Test. Danach müssen Sie als Host localhost oder Ihren Datenbankserver eintragen, den Login **root** eingeben. Klicken Sie auf OK. Wir haben nun dem Client alle Informationen gegeben, die er braucht, um sich mit der Datenbank zu verbinden. Damit er sich auch verbindet, klicken Sie auf das nun aktivierte Blitz-Icon. Nun werden plötzlich alle Icons aktiviert, und bei **Test** erscheint ein [+]. Klicken Sie nun auf das [+]. Es erscheinen nun als Festplatten die verschiedenen Datenbanken. Nur unter **mysql** erscheint wieder ein [+]. Klicken Sie nun dort drauf. Nun sehen Sie als blau-weisse Icons dargestellte Tabellen.

Klicken Sie nun mit der rechten Maustaste auf eine der Tabellen. Es öffnen sich mehrere Menüs. Klicken Sie nun auf **Describe Table**. Es erscheint nun ein neues, großes Fenster mit alle Spalten und Definitionen:

```
Field           Type              Null    Key     Default Extra
Host            char(60)                  PRI
Db              char(64)                  PRI
User            char(16)                  PRI
Select_priv     enum('N','Y')             N
....
....
```

Wir sehen nun wieder alle aus dem originalen MySQL Manager angegebenen Tabellendefinitionen und die Grant-Tabellen für die Angabe der Benutzerrechte. Zu deren Bedeutung sehen Sie bitte unter GRANT in der Sprach-Referenz von **MySQL** nach. Spielen Sie nun mit den Menüs herum. Sie können hiermit einfach SQL-Statements absetzen (select * from user), die Datenbank in eine Datei exportieren [Databases]->[Drop Database], oder auch Daten aus anderen Datenbanken auf Ihre Arbeitsstation übertragen und umgekehrt. Daten, die Sie z.B. auf Ihrer Arbeitsstation eingegeben haben, können Sie nun nach folgendem Schema auf eine MySQL Datenbank (oder auch eine andere) im Internet exportieren:

1. Starten Sie den MySqlWinAdmin.
2. Stellen Sie eine Verbindung zum Internet her. (ISDN)
3. Konnektieren Sie sich an die SQL-Datenbank auf Ihrer Arbeitsstation.
4. Dassselbe tun Sie nun mit Ihrer Datenbank im Internet. (Beide müssen identische Strukturen haben)
5. Nun kopieren Sie mit SELECT .... alle Daten von Ihrer Datenbank auf die Datenbank im

Internet. Die Daten stehen sofort dann im Internet zur Verfügung.

6. Falls Sie mehrere Tabellen haben, die koinsistent sein müssen, so kopieren Sie auch diese Tabellen, und reinitialisieren den MySQL Server im Internet neu.

An dieser Stelle haben wir natürlich noch keine Datenbankstrukturen angelegt, sodaß auch kein SELECT Statement angegeben werden kann. Zu einem späteren Zeitpunkt werden wir diese Prozedur im Detail beschreiben. Prinzipiell wissen Sie aber nun, wie es gehen kann. Es gibt aber noch viel mehr Möglichkeiten, Datenbanken zu kopieren.

Spielen Sie mit dem Manager ein wenig herum, und versuchen Sie anhand der MySQL Sprachreferenz zu ergründen, was die **Table Keys** alle bedeuten, und wofür diese gut sein könnten ;-)






54.1 Der phpMyAdmin Manager

Es gibt nichts, was nicht zu verbessern wäre. Unter PHP 3 gibt es nun den PHPMyAdmin Manager für MySQL Datenbanken, der einem wirklich alle umständlichen Arbeiten mit den Tools unter der DOS/UNIX-Shell abnimmt. Es läßt sich sogar über das Internet der komplette MySQL-Server administrieren. Dies ist das Toolkit, welches wird in leicht modifizierter Version auch auf unseren Datenbank-Servern einsetzen. Für die Vertreter der Klick Generation unter Windows läßt sich dieser natürlich auch auf Ihrer Arbeitsstation für die Administration des **MySQL** Servers einsetzen. Voraussetzung ist allerdings, daß der Apache WWW-Server 1.3.6 und PHP 3.0 nach der Anleitung hier im Handbuch installiert wurden.

Eigenschaften des phpMyAdmin Mangers

- Datenbanken anlegen und löschen
- Anlegen, Kopieren, löschen und verändern von Tabellen
- Löschen, editieren und hinzufügen von Feldern
- SQL Statements und Batch-Statements starten
- Keys in Feldern managen
- Text-Files in MySQL importieren
- Table Dumps speichern und einlesen
- Exportieren von Daten in das CSV-Format für z.B. Excel
- Administrieren mehrerer Server und auch einzelnen Datenbanken
- Kaffe kochen kann er nicht ;-(

Lesen Sie hierzu auch das Tutorial Einsteiger Tutorial für MySQL unter LINUX, welches Einsteigern eine einfache Möglichkeit bietet, MySQL professionell zu administrieren.

Installation des phpMyAdmin Mangers

Laden Sie sich das PHP-Script (es ist sehr klein) von dem Server http://phpwizard.net/phpMyAdmin/. Grundvoraussetzung ist, daß PHP 3.0 auch interpretiert werden kann. Hierzu installieren Sie bitte die WIN 32 Version des Apache WWW-Servers und PHP genau nach der Anleitung hier im Handbuch. Der IIS-Server kann zwar PHP 3.0 ebenfalls interpretieren, die Microsoft Software ist aber gegenüber den herausragenden Fähigkeiten des Apache WWW-Servers einfach hinter dem Mond. Vernünftiger Support von Microsoft ist auch nur für 10.000 US$/Jahr zu bekommen. Die Quellcodes gibt es für ca. 300.000 US$, eine Garage ist für die CDROM's freizuhalten. Außerdem sind inzwischen 1.2 Millionen Apache WWW-Server im Internet installiert, dagegen ist Microsoft eine Klitsche. Es gibt aber noch einen wichtigen Grund, den Apache WWW-Server einzusetzen. Die Beispiele sind dann auch auf unserem WWW-Server ohne Änderungen lauffähig.

Die Datei phpmyadmin.zip extrahieren Sie bitte in das Verzeichnis **c:\apache\htdocs**. Dort wird dann automatisch ein Verzeichnis namens **phpMyAdmin** angelegt. Fertig !

Ihr Browser sollte nun auf die URL http://localhost/phpmyadmin/index.php3 zeigen, um die lokale MySQL Datenbank administrieren zu können. Sie können nun eine neue Datenbank anlegen und mit den zuvor genannten Administrationswerkzeugen überprüfen, was passiert ist. Sie können alle Administrationswerkzeuge ohne Bedenken gemischt einsetzen.

Das schöne an PHP 3 ist, daß es überall funktioniert, sogar auf Windows 98, Windows NT, Apple Macintosh, allen Derivaten von UNIX, OpenVMS, u.s.w.

Sie haben nun erfolgreich das Supertoolkit für MySQL Datenbanken (und auch PostgreSQL) installiert, mit welchem sogar Sie als Einsteiger (oder als Fortgeschrittener) einfach SQL-Datenbanken anlegen und vollständig administrieren können.

Eine Besonderheit sollten Sie sich vielleich merken: Sie können auch MySQL Server über ein Netzwerk administrieren, wobei der APACHE Server mit PHP auf Ihrer lokalen Arbeitsstation läuft.

Insgesamt kann man sagen, daß es kein besseres Werkzeug für die MySQL Datenbank gibt, weil es wirklich alle Eigenschaften von MySQL **voll** unterstützt.

Machen Sie sich mit diesem Werkzeug auf Ihrer Windows Arbeitsstation vertraut, und spielen Sie ein wenig herum. Legen Sie Datenbanken an, erzeugen Sie Tabellen, ändern Sie Tabellenstrukturen. Nur vor **DROP** sollte Sie sich hüten, weil man hiermit Felder und andere Dinge in der SQL-Datenbank löscht. Sie können von Ihrer Arbeitsstation ASCII-Dateien z.B. aus Serienbrief-Dateien auf Ihre MySQL-Datenbank über das Netzwerk übertragen. Dasselbe gibt natürlich auch für Ihre Datenbank im Internet.








55. MySQL und ACCESS - Vorüberlegungen

ACCESS 97 ist ein in Unternehmen gerne eingesetztes Werkzeug. Was ACCESS 97 besonders auszeichnet, ist die einfache Erstellung von Datenbanken und vor allem der Ein-und Ausgabeformulare.

Viele Datenbanken werden zuerst in ACCESS entworfen, um dann später durch einen SQL-Server ergänzt zu werden. ACCESS hat leider die unangenehme Angewohnheit, daß bei jedem Suchlauf die gesamte Datenbankdatei linear auf der Festplatte durchsucht wird. Für eine einzelne Arbeitsstation ist dies kein Problem. Wenn aber mehrere User gleichzeitig auf die Datenbank zugreifen möchten, dann ist es zwingend notwendig, daß die Datenbankdatei auf einem Fileserver im Netzwerk abgelegt wird. Dieser könnte durchaus auch eine Arbeitsstation mit einer Freigabe für WfW Clients oder ein LINUX Server mit SAMBA sein. Problematisch wird dann die Netzwerklast, wenn jeder Datenbank-Client für eine Suchabfrage die komplette ACCESS Datenbankdatei über das Netzwerk transferiert. Dies führt u.U. zu massiven Störungen im Netzwerk bei anderen Anwendern. Hier hilft eine SQL Datenbank, die zentral alle Daten enthält, und auf welche die Clients mit ACCESS 97 als Frontend, also als Benutzeroberfläche zugreifen. Es werden dann nur noch diejenigen Daten über das Netzwerk transferiert, die der Anwender angefordert hat. Diese werden dann von ACCESS 97 dargestellt. Für den Anwender gibt es dann keinen Unterschied zu der Lösung ohne SQL Datenbank, außer - daß die Suche erheblich beschleunigt wird. Der Geschwindigkeitsvorteil fällt dabei besonders dramatisch aus, wenn die Datenbank sehr groß ist, oder viele Anwender gleichzeitig auf die Daten zugreifen möchten.

Bedenken beim Einsatz von ACCESS als Frontend und MySQL als Backend haben sich als unbegründet erwiesen. Zahlreiche Unternehmen setzen diese Kombination seit Jahren ! ein. Das Bestreben der Entwickler war es auch stets, eine stabile Client - Server Lösung mit MySQL zu garantieren. Support ist natürlich ebenfalls gegeben. Falls Sie sich für verteilte Datenbanken interessieren, dann möchte ich Ihnen das Kapitel Verteilte Datenbanken mit MySQL empfehlen.







55.1 Überlegungen zur Performance

Man kann sich leicht überlegen, daß bei einer ACCESS Datenbank nicht mehr der Prozessor, sondern eher die Festplatte das Nadelöhr ist. Wir gehen von einer kleinen Datenbank mit ca. 300 Einträgen aus, die gerade noch auf eine Diskette passt. Die ACCESS Datenbankdatei enthält neben den Daten viel Ballast, wie Alle Eingabeformulare, Masken u.s.w., sodaß schon die kleinste Datenbank ca. 2 MByte Größe erreicht. Bei einer typischen Duchsatzrate von 5 MByte/Sekunde zur Festplatte dauert ein Suchlauf also ca. eine halbe Sekunde. Bei 10.000 Einträgen und angenommenen 10 MByte Größe der Datenbankdatei muß dann schon mit Zeiten von 2-3 Sekunden gerechnet werden.

Problematisch wird es, wenn die Daten über das Netzwerk übertragen werden. Ein 10 MBit LAN mit einer Nettorate von ca. 600 KByte/Sekunde, die typisch für Microsoft Netzwerke ist, wäre also ca. 6 Sekunden mit dem Transfer der ACCESS Datenbankdatei beschäftigt. Für 6 Sekunden ist das LAN dann bis zur Leistungsgrenze ausgelastet. Katastrophal wird das Antwortverhalten, wenn mehrere Anwender gleichzeitig in der Datenbank recherchieren. ACCESS unterstützt allerdings die Möglichkeit, die Datenbank zu teilen, und die Datensätzte von Zeit zu Zeit abzugleichen, bzw. zu synchronisieren. Hierzu sehen Sie bitte im Handbuch ACCESS 97 nach. Was uns jedoch mehr interessiert, ist die Performance bei der Anbindung von SQL Datenbanken.

Die SQL Datenbanken liegen mit ihrer Performance allesamt recht gut im Rennen, da die Datenbankdatei bei SQL nicht wie bei ACCESS linear durchsucht wird. Stattdessen erfolgt der Zugriff auf die Daten intern mit Hilfe einer Baumstruktur (z.B. B-Tree, AVL-Tree...). Die Suchzeit ist also nicht linear zur Größe des Datenbankfiles, sondern günstigstenfalls proportional zum Logarithmus der Einträge.

Stellen Sie sich hierzu einen Baum vor mit jeweils 26 Verzweigungen für jeden Buchstaben des Alphabetes und überlegen Sie, wieviel schneller sie z.B. Ihren Namen finden, wenn Sie sich einfach der Reihenfolge der Buchstaben entlang der Äste hangeln. Im Vergleich zu einer linearen Liste finden Sie Ihre Einträge sehr viel schneller. ACCESS hingegen durchsucht stets die gesamte Datenbankdatei, auch wenn intern die Daten als Baum abgelegt sind.







55.2 MySQL - Performancevergleich

Wer sich für Vergleichswerte von MySQL zu anderen Datenbanken interessiert, der mag auf der Homepage von MySQL nachschauen: http://www.tcx.se. MySQL ist wesentlich schneller als andere SQL Datenbanken. Das liegt daran, daß Präprozessoren, wie Transaktionsmanager, Logmanager u.s.w. entfallen. MySQL besitzt kaum Overhead. Die Performance ist wirklich nur von der Hardware abhängig. MySQL 3.23 (die neue Version) reizt die Hardware bis an Ihre Leistungsgrenze aus, während hingegen alle anderen Datenbanken die CPU erheblich belasten, und die das Festplatten Interface nie voll ausnutzen. Siehe Zeitschrift i´X 8/99. Bei MySQL können Sie durch die Wahl der Festplatte und des Interfaces fast genau vorhersagen, wie schnell Ihre Daten gefunden werden.

Nebeneffekte, die eines Feintunings bedürfen, kennt MySQL also nicht. Den Benchmark SQLBench2 von http://www.benchmark-resources.com absolviert MySQL auf einem Pentium 450 mit 512 MByte RAM und Hardware RAID 5 unter LINUX in ca. 0.4 Sekunden. Oracle liegt hier bei 13 Sekunden, Sybase bei 7 Sekunden, Informix bei 24 Sekunden. Man muß allerdings dazu sagen, daß dieser Benchmark nur die Performance bei Lesezugriffen testet. Das Verhalten bei Schreibzugriffen vieler User gleichzeitig ist bei MySQL oft nicht vorhersagbar, da MySQL eine andere Art von Locking besitzt, als die namhaften Datenbankhersteller. Langsam ist MySQL auch hier keinesfalls.







55.3 Performance der ODBC-Treiber

Microsoft ODBC Treiber haben den Vorteil, sich inzwischen zu einem Standard etabliert zu haben. Auch unter UNIX gibt es inzwischen für alle Datenbanken freie ODBC-Treiber, die dann StarOffice oder Applixware Office den Zugriff ermöglichen. Man sollte vielleicht meinen, daß mit Hilfe des ODBC-Treibers zwischen Anwendung und SQL-Server die Daten effektiv übertragen werden. Dem ist leider nicht so. Microsoft hat in die ODBC-Pakete eine Menge Ballast eingefügt. So werden z.B. bei jedem absetzen einer SQL-Anfrage von ACCESS an die SQL-Datenbank im Netz alle Informationen über die Spaltentypen mit übertragen, damit die ODBC-Treiber eventuell Formate anpassen und konvertieren können. Z.B. existert zwar bei allen Datenbanken der Datentyp DATUM, ACCESS hätte es aber gerne als STRING von CHAR (VARCHAR) geliefert. Die Anpassung der Datentypen von der SQL-Datenbank zur Anwendung regelt der ODBC-Treiber. Im LAN erfährt man mit ODBC und SQL eine erhebliche Performanceverbesserung der ACCESS-Datenbank. Bei der Übertragung der Daten über ISDN kann man feststellen, daß die Grenze mit ca. 20-30 simultanen Clients erreicht ist. Wer also z.B. über eine ISDN-Standleitung an das Internet angeschlüssen ist, und Daten für Kunden über Internet via ODBC-Treiber anbieten möchte, der sollte vielleicht die JDBC KLasse4 - Treiber, die keinen Ballast mit übertragen, vorziehen. Beispiele zur Entwicklung von JDBC-Treibern finden Sie auf unseren Support Seiten







55.4 Migration zu Microsoft SQL

Microsoft bietet hier zur einfachen Migration ein Toolkit an, welches die Tabellenstrukturen aus ACCESS ausliest, und in die DDL (Data Definition Language) für die indentischen Microsoft SQL oder Sybase SQL Server umwandelt. Danach muß der ODBC-Treiber installiert werden, über welchen dann alle Anwendungen auf die SQL-Datenbank zugreifen können. Alle ACCESS Tabellen werden dann auf den SQL-Server übertragen. Danach muß die ACCESS Datenbank auf Abfrage des SQL-Server umgestellt werden. In fast allen Fällen ist es jedoch notwendig, die Datenbankstruktur und die SQL-Statements noch einmal zu überdenken und die Abfragen anders zu gestalten. Wenn man z.B. aus einer Datenbank alle Kunden einer Stadt, die mehr als 50.000 DM umgesetzt haben suchen möchte, so kann man dies auf 2 Arten tun. Einmal läßt man die SQL-Datenbank die **und** Verknüfpung durchführen und danach die Daten übertragen, oder man läßt ACCESS 97 die Verknüpfung durchführen. Im zweiten Fall ist die Netzwerklast enorm, da ja ACCESS zuerst alle einträge der Kunden einer Stadt erhält, danach alle Kunden mit einem Umsatz von mehr als 50.000 DM. Nachdem viele überflüssige Daten über das Netzwerk übertragen worden sind werden diese mit **und** verknüpft und an den Anwender ausgegeben. Damit solche Dinge nicht passieren, muß man ohnehin die Ergebnisse des Konverters noch einmal nachbearbeiten. Aber auch für MySQL gibt es einen Access2MySQL Konverter.







55.5 Typische Probleme beim Konvertieren der ACCESS Datenbank

Zur Migration einer ACCESS Datenbank auf MySQL steht ein Konverter Access2MySQL zur Verfügung, mit welchem man die **CREATE TABLE....INSERT ...** Befehle für MySQL automatisch ausgeben lassen kann. Dieser läuft als ACCESS Makro und ist leicht zu bedienen. Problematischer sind aus meiner Erfahrung falsche Datenbankstrukturen, die bereinigt werden müssen. Gerade Einsteiger mit ACCESS benutzen gerne JA/NEIN Felder, um Objekten bestimmte Eigenschaften zuzuordnen, ein typischer Fehler:

```
+----+------------------------+------------------------+------------------------+
|Name| Haarfarbe rot (JA/NEIN)|Haarfarbe blond (JA/NEIN)|Haarfarbe braun (JA/NEIN)|
+----+------------------------+------------------------+------------------------+
```

Hier werden zur Beschreibung von einer Eigenschaft gleich drei boolsche Felder angelegt, die dann nur mit JA/NEIN Informationen gefüllt sind. Besser wäre hier der Typ TEXT in Verbindung mit einem Listenfeld oder Kombinationsfeld in der Eingabemaske korrekt. Allein durch das Hinzufügen der Haarfarbe schwarz oder weiß müßte die ganze Struktur von ACCESS und der SQL Datenbank verändert werden. Bei solchen struktuellen Fehlern kann kein Konverter mehr helfen. Lesen Sie sich hierzu das Kapitel ER Diagramme durch.







55.6 Variablentypen unter ACCESS und MySQL

Bei der Adaption der ACCESS Datenbank an die Datentypen von MySQL müssen einige Dinge beachtet werden. Erstens gibt es Datentypen in ACCESS, die in MySQL sich mit dieser Bezeichnung so nicht wiederfinden. Andererseits scheint MySQL Datentypen zu unterstützen, die ACCESS wiederum nicht unterstützt. Schauen wir uns einmal die Datentypen von ACCESS 97 einmal an:

1. **TEXT** ist ein Feld für Buchstaben und Zahlen, mit welchen aber nicht gerechnet werden kann. Beispiel: 02227/80768 oder Guido Stepken
2. **MEMO** erlaubt den Eintrag von Texten mit bis zu 32.000 Buchstaben, Zahlen und Sonderzeichen sind nicht erlaubt.
3. **Zahl** enthält nur Zahlen, mit denen man später auch rechnen können muß. Diese können 1,2,4,8 Byte lang sein.
4. **Datum/Zeit** enthält Angaben über Datum oder Uhrzeit
5. **Währung** ist ein Feld mit intern 8 Byte, in welchem die Währung auf 2 Stellen gerundet wird.
6. **Zähler** kann pro Tabelle nur einmal vergeben werden und zählt automatisch hoch.
7. **JA/NEIN** ist ein Bitfeld, oder auch boolsches Feld genannt, in welchem man z.B. Eigenschaften, wie Mitglied oder Nichtmitglied speichern kann.
8. **OLE Objekt** kann eine Grafik, ein Bild oder auch irgendein binäres Objekt sein, welches bis zu 128 MByte groß sein darf.

Die folgende Tabelle enthält eine Liste der ANSI SQL-Datentypen und der dazu äquivalenten SQL-Datentypen von Microsoft ACCESS sowie die zulässigen Synonyme.

ANSI SQL Datentyp	Microsoft ACCESS Datentyp	Synonym
BIT, BIT VARYING	BINARY (Siehe Anmerkungen)	VARBINARY
Nicht unterstützt	BIT (JA/NEIN siehe Anmerkungen)	BOOLEAN, LOGICAL,LOGICAL1, YESNO
Nicht unterstützt	BYTE INTEGER1	
Nicht unterstützt	COUNTER AUTOINCREMENT	
Nicht unterstützt	CURRENCY (=Währung)	MONEY
DATE, TIME, TIMESTAMP	DATETIME (=DATUM/ZEIT)	DATE, TIME, TIMESTAMP
Nicht unterstützt	GUID	
DECIMAL	Nicht unterstützt	
REAL	SINGLE (=Zahl)	FLOAT4, IEEESINGLE, REAL
DOUBLE PRECISION,FLOAT	DOUBLE (=Zahl)	FLOAT, FLOAT8, IEEEDOUBLE, NUMBER, NUMERIC
SMALLINT	SHORT	INTEGER2, SMALLINT
INTEGER	LONG	INT, INTEGER, INTEGER4
INTERVAL	Nicht unterstützt	
Nicht unterstützt	LONGBINARY (=OLE-Objekt)	GENERAL, OLEOBJECT
Nicht unterstützt	LONGTEXT (=MEMO)	LONGCHAR, MEMO, NOTE
CHARACTER, CHARACTER VARYING	TEXT	ALPHANUMERIC, CHAR, CHARACTER,STRING, VARCHAR
Nicht unterstützt	VALUE	

Anmerkungen :

Der ANSI SQL-Datentyp BIT entspricht nicht dem SQL-Datentyp BIT von Microsoft Jet (ACCESS), sondern dem Datentyp BINARY.

Es gibt kein ANSI SQL-Äquivalent für den SQL-Datentyp BIT in Microsoft Jet. Das reservierte Wort VALUE stellt keinen der vom Microsoft ACCESS 97 definierten Datentypen dar.

Leider ist auch Microsoft selber zu allen SQL Standards inkompatibel, sodaß man hier sich bei MySQL am besten selber sein Äquivalent sucht. Das Problem dabei ist, daß man einen Datentyp aussuchen muß, der von dem ODBC-Treiber auch korrekt interpretiert oder durchgereicht wird. Hierzu gibt es bei ODBC einige Parameter, die sich justieren lassen, damit bestimmte

Datentypen automatisch konvertiert oder auch nicht konvertiert werden. Leider hat Microsoft je nach ACCESS Version hier abweichende Meinungen zu Datentypen, sodaß man kaum bestimmt sagen kann, welche Einstellungen bei ACCESS auch korrekt konvertiert werden. Korrekte Aussagen kann hier nur der Hersteller im Anschnitt über den ODBC-Treiber machen.

Insbesondere bei den Zahlen muß man sich rechtzeitig im Klaren sein, welche Werte abgespeichert werden, damit man die Datentypen nicht später auf mehr Byte erweitern muß. Mit Hilfe des Adminitrationstoolkits ist eine Änderung aber kein Problem mehr.

Für die Speicherung von BOOL´schen Datentypen (JA/NEIN) unter ACCESS eignet sich in MySQL der Datentyp **ENUM**








55.7 Verknüpfte Tabellen mit MySQL

Die Verknüpfungen von Tabellen in ACCESS sind recht einfach zu erstellen. Die Verknüpfungen ähneln den **foreign keys** unter SQL. Das hat zur Folge, daß bei der Löschung einer Zeile aus einer Tabelle automatisch die korrespondierende Zeile aus einer oder mehreren anderen Tabellen gelöscht wird. Bei der Konvertierung von ACCESS Datenbanken nach MySQL mit dem Migrationstoolkit können diese Abhängigkeiten der Tabellen natürlich nicht mit übertragen werden, da MySQL keine **foreign keys** unterstützt. Hier kann man sich aber mit den Vorschlägen aus dem Abschnitt Workarounds für SQL Statements behelfen. Das bedeutet, daß man sich mit SQL auskennen muß, um diese Workarounds auch anwenden zu können. Aber auch bei dem Migrationstoolkit von Microsoft selber läuft nicht alles reibungslos, wie man auf der Site http://www.ntfaq.com/sql/ sehen kann. Man kommt also insgesamt nicht um das Erlernen von SQL herum. Schauen Sie hierzu in das Kapitel MySQL Tutorial und lernen Sie die Schritt für Schritt die Sprache SQL.**********







55.8 Eingabemasken von ACCESS 97

ACCESS bietet die Möglichkeit, bei der Eingabe der Daten bereits Menü´s zu erstellen, die sich am Inhalt des Datenfeldes orientieren. Dazu wird jeweils eine separate Abfrage erstellt, die dann im Eingabeformular dargestellt wird. Diese Abfrage ist in SQL geschrieben und läßt sich ohne Probleme 1:1 übernehmen. Es wird dann bei jeder Eingabe ein SELECT Statement an MySQL übergeben, damit die Auswahl für das Feld dargestellt werden kann. Man findet es in der Toolbox als Kombinationsbox. Es dürfte dann aber klar sein, daß bei jeder Eingabe die SQL Datenbank zum Aufbau des Pulldownmenüs in der Eingabemaske abgefragt wird, was einigen Streß auf dem Netzwerk bedeutet. Da diese Abfragen aber sehr kurz sind, fallen diese nicht weiter auf.







55.9 Replikationen von Datenbanken unter ACCESS

ACCESS erlaubt die Replikation von Tabellen, sodaß dezentral Daten erfasst werden können. Später kann dann über DFÜ ein Abgleich der Daten erfolgen. Um ACCESS hierzu vorzubereiten, muß eine Design Master Datenbank (DMR) angelegt werden, in welcher festgehalten ist, welche Datenbank die Master und welche die Client Datenbanken sind. Nachdem alles korrekt läuft, muß die Master Datenbank an die MySQL Datenbank mittels ODBC-Treiber angebunden werden. So fließen dann alle Änderungen der dezentral erfassten Daten über den Synchronisationsmechanismus von ACCESS in die zentrale SQL Datenbank des Unternehmens und zurück. Eine ideale Lösung für Filialen, Vertreter u.s.w. Über das Internet können sich diese dann über einen mit PPTP verschlüsselten Kanal mit der SQL Datenbank des Unternehmens synchronisieren.







56. Frontends und Cursor unter MySQL

Cursor erhöhen bei Anwndungsprogrammen erheblich den Komfort bei der Datenerfassung und der Änderung. In vielen Fällen sind diese einfach unentbehrlich. Man findet häufig die Aussage, daß Cursor unter MySQL nicht unterstützt werden. Das ist zwar richtig, aber man muß auch dazu sagen, daß Cursor und deren Implementierung noch Fossile aus der Vergangenheit sind, wo Mainframes mit ThinClients noch Standard waren. Heutzutage gibt es kaum noch ThinClients, sodaß z.B. Serverside Cursor nicht mehr notwendig sind. Mit Hilfe von ThickClients, also normalen PC´s unter Microsoft Windows kann man völlig ohne **serverside cursors** komfortabelste Anwendungen schreiben. Es ist inzwischen völlig egal, welche Datenbank hinter den Frontends (ACCESS, JAVA RMI....JDBC, ODBC) steht, da die intelligenten Frontends z.B. Cursor und Caching selber verwalten. Die SQL Datenbank steuert nur die durch SELECT angeforderten **result sets** bei und übergibt diese an den Client. Probleme, wie Scrollig u.s.w. übernimmt der das Frontend. Allen voran ist ACCESS sicher dasjenige Frontend, die es einem Anwender ohne Programmierkenntnissen es ermöglicht, eigene Frontends für Datenbanken zu erstellen (Ich selber habe ca. 1 Tag Einarbeitung gebraucht ...).

Verfechter der individuell programmierten Frontends, die von den komplexen **serverside cursors** Gebrauch machen, werden immer einsamer.

Cursor sind eine komplexe Angelegenheit. Es gibt Serverside, Clientside, Forward-Only .... Cursor. Danach ist es eine Frage des Anwendungs-Interfaces, also den Fähigkeiten des ODBC-Treibers oder JDBC Treibers. MyODBC unterstützt in Verbindung mit Microsoft Applikationen **clientside cursors**, auch als **front-end cursors** beschrieben. Diese funktionieren mit allen Microsoft Anwendungen und Programmiersprachen. Serverside Cursor werden von MySQL nicht unterstützt, da diese oft erhebliche Performanceprobleme auf dem Server machen. Bei vielen simultanen Clients, wie sie oft im Internet auftreten können, sind Serverside Cursor das Ende jeder SQL Datenbank. Der Grund dafür liegt einfach darin, daß alle Statements über einen Postprozessor der Datenbank laufen, der dann die Cursor mit der Ausgabe der Statements verknüpft, und daraufhin entscheidet, welche Daten dem Client übermittelt werden. Das kostet Auslagerungsspeicher und viel CPU-Zeit. Clientside Cursor schonen erheblich Serverresourcen, da diese neuerdings Caching - Mechanismen unterstützen. Bei Internet-Anwendungen sind also in jedem Falle eine hochkarätige Datenbank, wie z.B. ORACLE, SYSBASE oder MS-SQL mit **serverside cursors** völlig ungeeignet. Unter JDBC gibt es ebenfalls **clientside cursor** für MySQL, die auch völlig problemlos funktionieren. Einige Hersteller bieten für MySQL erweiterte Bibliotheken an, die erhebliche Erweiterungen und Vorteile gegenüber den RMI JDBC Bibliotheken von SUN bieten.







56.1 Seitenweise Ausgabe von Daten

Die einfachste Art, Daten seitenweise auszugeben, ist es, die Zahl der ausgegebenen Tupel (Zeilen) der MySQL Datenbank zu begrenzen. Einige SQL Spezialisten schwören auf Cursor, obwohl es doch recht einfach auch so funktioniert:

```
SELECT * from table LIMIT ab_eintrag_nummer, zahl_der_eintraege;
```

Allerdings gibt es noch einen Trick. Man speichert sich das letzte Tupel in einer Variablen, und läßt sich dann alle Einträge nach dieser Variablen ausgeben. Ein Beispiel:

```
SELECT * from table where name >= 'der_zuletzt_ausgegebene_name' order by name
limit 20;
```

Mit diesen beiden Kniffen kann man fast alle Bedürfnisse von WWW-Interfaces abdecken, zumal die Sprache HTML echtes Scrolling unterstützt. Hierzu läßt man sich jeweils 50 oder 100 Datensätze in ein HTML Feld ausgeben, in welchem man dann die Daten, die man verändern will, markieren kann. Man nutzt als nicht mehr die Scrolling - Funktionen von SQL, sondern diejenigen des HTML Interfaces. Um z.B. das Zurückblättern zu realisieren, geht man genauso vor.







56.2 Cursor mit PHP3

Die Homesite von PHP3 läßt natürlich auch keine Wünsche offen, und zeigt anhand von Beispielen, wie man unter PHP3 Cursor für WWW-Frontends programmiert:
http://www.php.net/manual/ref.odbc.php3







56.3 Cursor mit PERL

Ein schönes Beispiel von Cursorn mit PERL auf einer SQL Datenbank, die Cursor unterstützt, findet man hier: http://linas.org/linux/db.html. Das Beispiel ist mit CSQ programiert und läßt sich nicht auf MySQL übertragen. Es dient nur dem Vergleich von MySQL mit Datenbanken, die Cursor unterstützen.







56.4 Funktionierende Cursor mit der C API unter MySQL

Siehe Handbuch MySQL C API Funktionen (in Arbeit)







56.5 Seitenweise Ausgabe mit ASCII Datenbanken

Eine Möglichkeit, sich seitenweise Daten aus einer ASCII Tabelle ausgeben zu lassen, soll an dieser Stelle ebenfalls nicht unerwähnt bleiben.

Beim Betrieb einer CDROM - Datenbank http://www.cd-special.de hat sich herausgestellt, daß SQL Datenbanken bei Datensätzen von fast 1.000.000 Einträge zwar sehr schnell sind, es jedoch einige User schafften, durch schnelles Blättern die Datenbank an die Grenze Ihrer Leistungsfähigkeit zu bringen. Daher sind wir den Weg der Zwischenspeicherung in ASCII Dateien gegangen. Nach einer Suche kann dann der User beliebig in den Datensätzen blättern, ohne daß die MySQL Datenbank in Anspruch genommen wird. Nach der Suche in der MySQL Datenbank werden die **result sets** über die Zuordnung der IP-Adressen des Clients im Internet einer ASCII Datei zugeordnet, in welcher dann der User in Ruhe blättern kann. Dieses läßt sich auch mit Cookies erreichen. Nach einiger Zeit werden diese temporären Dateien dann gelöscht. Hier ein Stück PERL Code, welches eigenständig so lauffähig ist:

```
#!/usr/bin/perl
#---------------------------------+
# Simple Database Engine 1.0      |
# by Pasha Golovko                |
# webmaster@find.virtualave.net   |
# http://find.virtualave.net      |
# More FREE scripts available at: |
# http://find.virtualave.net/00/  |
#---------------------------------+
#
# Chmod:
# database.cgi  755     (rwxr-xr-x)
# database.txt  777     (rwxrwxrwx)
# index.htm     777     (rwxrwxrwx)
#
# Display some header
$html1 = qq~
<html>
<head>
<title>My simple database

<body>
~;

# Display some footer
$html2 = qq~


~;

# Open my silly database
open(LOGFILE, "<database.txt");
@entries = <LOGFILE>;
close LOGFILE;


# Open the output file
```

```
open (HTML,">index.htm");


                # Print header
                print HTML"$html1";


# Now, lets split what ever we got in the database ...
foreach $line (@entries) {
        @fields = split(/\|/,$line);


                # ... and print what ever fields we got from database
                print HTML "$fields[0] - $fields[1] - $fields[2] -
$fields[3] <br>";
};
                # Print footer
                print HTML"$html2";
# Close the output file
close (HTML);
# Print some visual output, when your script finished working with database
:)
print "Content-type: text/plain\n\n";
print "Done!";
```








56.6 Cursor mit ACCESS und MySQL

Wer einmal ACCESS Formulare programmiert hat, der wird wissen, daß dieses Thema unter ACCESS einfach nicht so exisitert. Der Grund liegt darin, daß ACCESS bei jedem SELECT Statement die Datenbank-Datei vollständig linear durchsucht, und die Ausgabe in einem Puffer speichert, durch den der Anwender dann beliebig scrollen kann, ohne daß Daten neu übertragen werden müssen. Wir nun ACCESS an MySQL angebunden, dann werden die Select-Statements an die MySQL-Datenbank übergeben. Die Datensätze werden selektiert und dann an ACCESS entweder vollständig oder in kleineren Portionen (SELECT * FROM ...LIMIT ...) übergeben. Diese werden von ACCESS gecacht, sodaß der Anwender ohne einen weiteren Serverkontakt darin blättern und Datensätze verändern kann. Man kann es nicht laut genug sagen, ACCESS ist ein ideales Frontend für SQL Datenbanken, insbesondere auch insbesondere für MySQL. Probleme bei der Programmierung von Cursorn treten einfach nicht auf, da die schlichtweg überflüssig sind. Es gibt aber eine Ausnahme, die dann besonders auffällt, wenn das SELECT Statement eine riesige Menge an Daten an ACCESS (dem result set) zurück übermittelt. Damit die Größe des **result sets** begrenzt wird, muß in ACCESS an den sinnvollen Stellen das SELECT Statement durch **LIMIT ...** ergänzt werden.







56.7 Cursor mit Visual C++/BASIC und MySQL

Serverside cursors funktionieren mit MySQL, wie schon anfangs erwähnt, nicht. MySQL verfügt über keine Pre-oder Postprozessoren, die z.B. Operationen auf SETs und die Positionierung von Forward - und Backward Cursors regeln könnten.

Daher möchte ich mich hier besonders auf Beschreibung der oft überlegenen, und bei Microsoft gut dokumentierten **clientside cursors** oder auch **frontend cursors** genannten Cursor Lösungen konzentrieren.

Bis zur Version SQL Server 6.0, die im wesentlichen auf SYBASE SQL beruhte, konnte Microsoft keine Bibliotheken zum vorwärts und rückwärts Scrollen in einem `result set` anbieten. Die Folge war, daß Microsoft hierfür das Verhalten von Cursors in der `Client API` emulieren mußte. Diese Routinen finden sich in der **DBLibrary**. Diese läßt sich mit allen Visual Basic/C++, JAVA Programmiersprachen ansprechen. Inzwischen hat Microsoft das ODBC Interface entwickelt, welches Teile dieser Cursor Bibliotheken enthält.

Hier finden Sie eine Beschreibung über Cursor, so, wie Microsoft diese mit ODBC 3.0 / 2.5 (update) implementiert. Eine ausführliche Beschreibung für Visual Basic 5.0, ACCESS 97, EXCEL 97.... http://www.hipstream.force9.co.uk/dissertation/week5/ODBC.html ist ebenfalls enthalten.

Die DBLibrary kommuniziert gewöhnlich mit der SQL Datenbank, und erwartet das **result set** zurück, bei Microsoft als **TDS** (Tabular Data Stream) implementiert. Dieses **result set** wird dann von der DBLibrary gecacht, und an den Client übergeben. Der Client kann dann dieses **result set** bearbeiten, ändern, darin Daten löschen, u.s.w. Diese werden dann entweder auf Befehl oder nach einem Timeout auf den SQL Server zurückgeschrieben.

Dieses Caching hat allerdings auch einige Grenzen, die hier nicht verschwiegen werden sollen:

- Der (My)SQL Server hat keine Möglichkeit, die Locks selbstständig zu kontrollieren. Da die DBLibrary Daten blockweise abfragt, kann es passieren, daß mehr Datensätze gesperrt werden, als notwendig.
- Da Caching kann erhebliche Resourcen auf dem Client beanspruchen, wenn zu große **result sets** zurückgegeben werden.
- Bei großen Datenmengen kann es passieren, daß Daten über das Netzwerk an den Client übermittelt werden, die überhaupt nicht gebraucht werden. In diesem Falle würde aber auch eine serverseitige Lösung erhebliche Performanceprobleme bereiten. Es lohnt sich dann auf jeden Fall die SELECT ...FROM...LIMIT Statements nocheinmal zu anzuschauen.

Auf der Website http://www.pbc.ottawa.on.ca sind viele Beispiele zur Implementierung von Cursors mit Visual Basic und der Library **DBWiz** beschrieben.







56.8 Frontend mit ODBC

Alle Routinen der DBLibrary können auch über die ODBC Schnittstelle aufgerufen werden. Das ermöglicht, aus einer beliebigen Anwendung heraus, wie z.B. EXCEL, WINWORD, OUTLOOK, Internet Explorer ...über die ODBC Schnittstelle auf MySQL zuzugreifen. Somit lassen sich z.B. unternehmensweit alle Mail-Adressen oder Adresslisten in MySQL ablegen, ganz so, wie unter Microsoft BACKOFFICE, nur erheblich billiger. Hier nun eine ausführliche Beschreibung der Implementierung von ODBC Frontends mit ODBC 2.5/3.0 Treibern, so, wie Microsoft diese mit ODBC 3.0 / 2.5 (update) implementiert. Eine ausführliche Beschreibung für Visual Basic 5.0, ACCESS 97, EXCEL 97.... http://www.hipstream.force9.co.uk/dissertation/week5/ODBC.html ist ebenfalls enthalten.







56.9 Cursor Bibliotheken mit Python

Auch für die Programmiersprache PYTHON, die eine der mächtigsten Sprachen überhaupt ist, gibt es Cursor Bibliotheken: http://starship.skyport.net/~lemburg/mxODBC.html.







56.10 Cursor mit JDBC

Mit JAVA und JDBC Treibern lassen sich recht einfach Frontends zu Datenbanken schreiben, in denen man im Netscape Browser oder Internet Explorer in SQL Datenbanken Scrollen kann. Die Vor - und Nachteile sind im Kapitel Cursor mit JDBC unter MySQL ausführlich beschrieben.







56.11 Beispiele und ausgewählte Links !!!!!

In dem Kapitel Ausgewählte Links finden Sie Links mit ausführlichen Kommentaren zu den Inhalten dieser Webseiten und eine Angabe darüber, bei welchen Problemen Sie Teile der dort angeführten Beispiele Sie für eigene Zwecke gut verwenden können.







57. Ausgewählte Links

Die Diplomarbeit von Sascha Weber behandelt das Thema relationale Datenbanken etwas intensiver. Das Thema lautet: Bewertung und Vergleich des Einsatzes relationaler und objektorientierter Datenbanksysteme anhand eines Anwendungsbeispiels:
http://www.wiwi.uni-frankfurt.de/~sascha/diprtf.html

Hier eine ausführliche Abhandlung über Normalisierung von Datenbanken der GRADUATE SCHOOL OF LIBRARY AND INFORMATION SCIENCE in AUSTIN, Texas:
http://www.gslis.utexas.edu/~l384k11w/normstep.html

Eine allgemeine Abhandlung für Entscheider über Datenbank Interfaces....
http://nudes.photo.net/wtr/thebook/

Viele sehr gute PHP Beispiele .... http://www.phpwizard.net

Eine gute Site für PHP Probleme mit vielen Beispielen: http://www.php-center.de

Nette Beispiele für PHP: http://www.php-homepage.de

Allgemeine Abhandlung freier Datenbanken für Entscheider
http://www.cis.ohio-state.edu/hypertext/faq/usenet/databases/free-databases/faq.html

Beispiel für ein PHP Interface für Suchmaschinen http://linuxstart.com/documentation/howtos.html

Wie baue ich mir eine WWW-Datenbank Schritt für Schritt ?
http://www.sfu.ca/~yzhang/linux/webdb/toc.html

Eine hervorragendes Archiv zu PHP3 und MySQL MySQL Archiv für PHP 3

Hier weitere MySQL Hilfen !







57.1 ODBC Links

1. http://swt.wi-inf.uni-essen.de/~dstein/inas5/links.html Hier findet man viele Links..
2. Die original Microsoft Homepage über ODBC und die Ansteuerung von ODBC mit ADO, RDS, OLE DB, ODBC, MTS, COM, und DAO (was auch immer das alles ist), und einige unvollständige Tutorials zur Programmierung von ODBC mit C++. Diejenigen, die mit Hilfe des Microsoft ODBC SDK's ihren eigenen ODBC-Treiber schreiben, oder bestehende anpassen möchten, sind hier genau richtig.
3. http://www.roth.net/perl/odbc/faq/ beschreibt die Ansteuerung von ODBC Treibern unter PERL auf Microsoft Windows 98/NT. Hiermit lassen sich die SQL Datenbanken fast aller Hersteller in wenigen Zeilen PERL ansteuern.
4. http://users.ids.net/~bjepson/FreeODBC/ ist die Site für ODBC Treiberentwicklung unter GPL für UNIX und Windows. MIt diesen ODBC-Kits lassen sich für fast alle Betriebssysteme und Datenbanken ODBC Treiber entwickeln und anpassen.
5. http://metalab.unc.edu/freetds/ ist ein freies GPL Entwicklungstoolkit zur Ansteuerung von Microsoft SQL und SyBase SQL Datenbanken über TDS (Tabular Data Stream)
6. http://www.learnasp.com/ Alles, was man über ASP, ODBC und Datenbanken zur Anbindung von Microsoft Anwendungen wissen muß. Hier findet man unzählige Tutorials und Tips zu jedem Unterthema.







57.2 JDBC Programmierung und Treiber

1. http://www.gwe.co.uk/java/jdbc/ ist eine JDBC Treiber für MySQL mit kompletter Anleitung und Beispielen. Leider ist dieser etwas veraltet und nicht mehr supportet, funktioniert aber fein.
2. http://java.sun.com/products/jdbc/index.html Diese Seite von SUN selber enthält alles, was man zur Ansteuerung von JDBC Treibern unter JAVA (JDBC API) braucht. Allererste Wahl für Einsteiger.
3. http://www.sybase.com/products/internet/jconnect/ ist zwar die Site für Entwicklung von Anwendung von Applikationen unter JAVA für SYBASE, die Informationen gelten aber genauso für MS SQL und MySQL oder irgendeine andere SQL DAtenbank mit JDBC Treiber. Die dortigen Beispiele können fast ohne Änderung für MySQL übernommen werden.
4. http://www.kisco.co.kr/~hollobit/java/tutorial1/jdbc.html beschreibt die Ansteuerung von SQL Datenbanken über die ODBC-Schnittstelle mit JAVA. Hierzu hat Vijay Mukhi (Indien) eine Treiberklasse für ODBC unter JAVA geschrieben. Diese Site ist zwar völlig veraltet, aber sehr aufschlußreich.
5. http://www.thoughtinc.com/demos.html zeigt live im Internet einen JDBC Treiber. Das Beispiel zeigt, daß es offensichtlich auch möglich ist, tausende von Surfern minütlich über ein JDBC Interface an eine SQL Datenbank anzubinden .....







57.3 Datenbank Ansteuerung unter PERL

1. http://www.perl.com/pace/pub ist das Archiv für PERL, Dokumentationen und Tutorials schlechthin....
2. http://theory.uwinnipeg.ca/CPAN/data/DBI/DBI.html ist die komplette Beschreibung des DBI Interfaces zur Ansteuerung von SQL Datenbanken mit PERL. Er werden fast alle SQL Datenbanken unterstützt.
3. http://theory.uwinnipeg.ca/CPAN/by-category/07_Database_Interfaces.html beschreibt nochmals das DBI Interface, diesmal aber nach Kategorien geordnet.
4. http://www.oasis.leo.org/perl/exts/db/00-index.html Hier findet man die Software zur Ansteuerung von Datenbanken mit PERL. Es lohnt sich, das Archiv weiter zu durchstöbern.
5. http://www.pl.freebsd.org/ports/perl5.html ist das Archiv für PERL Software unter FreeBSD (oder auch NetBSD/OpenBSD) UNIX.
6. http://theory.uwinnipeg.ca/CPAN/by-name/ApacheDBI.html ist die Homesite für PERL Module im Apache WWW-Server. Mit diesen werden CGI-BIN's um ein vielfaches (Faktor 5-100) schneller ausgeführt.







57.4 MySQL Tutorials

1. http://www.devshed.com/Server_Side/MySQL/ ist eine interessante Site für MySQL Einsteiger. Hier findet man Support, Tutorials und Diskussionsgruppen zu vielen Unterthemen....







58. Projekt Kontakt-Datenbank

Projekt Kontakt-Datenbank mit MySQL und PHP von Mike Miller, Copyright Web Techniques







58.1 Beschreibung

Diese Kontaktdatenbank ist ein Adressbuch, welches schön zeigt, wie man mit zwei Tabellen arbeitet. Die Kontaktdatenbank kann einfach um viele Features erweitert werden.

Die Datenbank enthält zwei Tabellen, die Kontakt Tabelle, die die Adressen enthält, und einen Feld für die Kommentare, dem ID - Feld und eine Tabelle für Kommentare. Die Tabellendefinitionen 1a und 1b zeigen die Statements der DDL (Data Definition Language) von MySQL (oder auch jeder anderen SQL Variante) , mit der die Struktur der Datenbank im SQL Server generiert wird.

Die Beispiele 1c und 1d zeigen die Syntax, wie man diese Statements an den MySQL Datenbankserver sendet.

Alle Tabellendefinitionen entsprechen dem SQL Standard, mit einer Ausnahme: Das ID (uid) Feld ist ein `autoincrement` Feld. Auch wenn diese nicht zum Standard gehören, sind sie aber eine sinnvolle Erweiterung, die in alle SQL Datenbanken inzwischen unterstützt wird. Wenn also eine Reihe (oder auch Tupel genannt) angefügt wird, dann wird eine anschließende, laufende Nummer vergeben. Es ist immer nur ein `autoincrement` Feld pro Tabelle zulässig. Ohne diese Erweiterung müßte man ersatzweise das Statement so schreiben:

"INSERT INTO contact VALUES (SELECT(max(uid) +1 FROM contact..."

Das Problem dabei ist, daß man immer ein Subselect ausführen muß, welches von MySQL aber nicht unterstützt wird. Eine andere Möglichkeit wäre es, die Kontakt Tabelle zu sperren, die maximale UID zu bestimmen, und den neuen Kontakt dann in UID +1 einzufügen.

Das C API Interface, mit welchem man MySQL Datenbanken aus C++ heraus ansteuern kann, sieht hingegen diese Möglichkeit vor. Die Funktion heißt **mysql_insert_id()**. Mit Hilfe von PERL ist dies ebenfalls möglich, das Konstrukt würde dann **$contact->insert_id** heißen.

In dem letzten Teil der Tabellendefinition sind KEY Werte definiert. Diese INDEX Attribute sollten stets bei Feldern hinzugefügt werden, die viel abgefragt werden. Sie beschleunigen die Suche in der Datenbank erheblich, insbesondere auch bei verknüpften Tabellen. Da UID ein `autoincrement feld` ist, muß es als **PRIMARY KEY** definiert werden. Der User sieht dieses Feld nicht, es dient nur der eindeutigen Verknüpfung mit einem Kontakt. In der anderen Tabelle wird dieses Feld analog zugeordnet (KEY uid). UID ist dasjenige Feld, was den Kontakteintrag mit der Kommentar Tabelle verknüpft. In der Kommentar Tabelle ist dieses Feld nicht eindeutig vergeben (unique), was bedeutet, daß zu einem Kontakt Feld gleich mehrere Kommentare zugeordnet werden können. Das nennt man **one-to-many relationship**. Das ist auch der Hauptgrund, warum diese Datenbank in zwei Tabellen aufgeteilt wurde. Siehe hierzu auch das Kapitel Entity Relationship. Die Datenbank enthält zudem noch **contact_date** und **contact_comment** Felder, beide sind INDEX Felder, damit die Suche und damit die Verknüpfung schneller erfolgt. INDEX Felder sind in Wirklichkeit HASH-Tables und deswegen rasendschnell, nebenher gesagt.







58.2 Das WWW-Interface

Nun ist die Datenbank unter MySQL definiert. Mit Hilfe des PHP3 Interfaces kann man nun über den Apache WWW-Server auf die Datenbank zugreifen. Das fertige Skript muß man nun einfach in das Homeverzeichnis des WWW-Servers kopieren. Das war´s im Prinzip auch schon.

Das Skript enthält eine **switch** Anweisung am Ende, die überprüft, welchen Status die verdeckte Variable **$state** besitzt. Dieser Status variiert in Abhängigkeit der vorhergehenden Aktionen. Beim ersten Aufruf wird eine Funktion namens **Main_Menu** aufgerufen, welches eine Reihe von **RADIO** Buttons zeigt, welches es dem User erlaubt, auszuwählen, ob er nun einen neuen Kontakt anlegen möchte, oder ob er Kontakte anzeigen möchte.

Wenn er sich für die Anzeige entscheidet, dann wird das Skript erneut aufgerufen, diesmal aber mit der Variablen **$state** auf `list` gesetzt. Das führt dazu, daß die List Funktion aufgerufen wird.

Hier sehen wir nun die Abfrage des SQL Servers mit PHP.

Die Abfrage ist einfach zu verstehen. Zuerst wird eine Verbindung zu dem Server aufgebaut. Bis dahin ist noch keine Datenbank ausgewählt. PHP fordert, daß die Datenbank genau angegeben wird, schließlich kann ein MySQL Datenbankserver bis über 1000 Datenbanken und Tabellen verwalten.

Die nächste Zeile des PHP Skriptes zeigt das. Die MySQL Funktion bindet sich an eine Datenbank mit dem ersten Parameter und führt das Kommando, welches als zweiter Parameter angegeben ist, aus. Danach speichert es alle von SQL zurückgegebenen Reihen ab, und ordnet eine Variable **$result** als Zeiger auf die zurückgegebenen Werte zu.

Der Standard, wie man nun an die Werte kommt, ist eine Schleife zu konstruieren, die über alle zurückgegebenen Zeilen läuft, und mit Hilfe der Funktion **mysql_result** die Werte ausliest.

Wie man sehen kann, ist das alles recht einfach zu verstehen, da sich PHP3 an BASIC anlehnt.

Nachdem der User die Informationen in sein Formular eingegeben hat, wird die Funktion **Commit_Contact** aufgerufen, die die Daten dann in die Datenbank einfügt. Nachem die Anbindung an die Datenbank hergestellt ist, wird ein SQL Statement abgesetzt, welches das aktuelle Datum und die Uhrzeit übermittelt, und einen Kommentar erzeugt, wann der Kontakt eingegeben wurde.

Hierzu wird die Funktion **mysql_insert_id** verwendet, die dem Kontakt einen eindeutigen Wert zuordnet, damit man mit einer Abfrage auch nach der ID suchen kann, die einem neuen Kontakt zugeordnet wurde.

In SQL müssen String Typen immer "gequotet" werden, Zahlen hingegen nicht.

Das größte Problem bei MySQL Datenbanken ist es, die Rechte für die Zugangskontrolle zur Datenbank zu vergeben. Sie enthält drei Tabellen: User, DB und Host. Die wichtigste Tabelle ist die User Tabelle. Die anderen kann man in der Praxis vernachlässigen und leer lassen, ohne zumindest einen Eintrag in die User Datenbank ist ein Zugriff auf die neu angelegte MySQL Datenbank nicht möglich.

Die Dokumentation grant enthält eine ausführliche Anleitung, wie man diese Rechte anlegt. Hier ein Tip: Vergessen Sie die Host Tabelle und setzen Sie alle Host Felder auf %, welches in SQL als Wildcard Buchstabe gilt, und auf alle Strings zutrifft. Das Feld Passwort darf während des Aufbaues der Datenbank ebenfalls leer bleiben.

Die Privilegien, die in der User Tabelle angegeben werden, setzen sich über diejenigen hinweg, die in der DB Tabelle eventuell schon entzogen werden, weil diese mit ODER Verknüpft werden. Man sollte alle Privilegien in der User Tabelle auf N setzen, um sicherzugehen, daß man nur Zugriffe von Datenbank zu Datenbank erlaubt hat.

Wenn die Rechte aufgesetzt sind, empfielt es sich, mit **mysqladmin reload** am Prompt des **mysql** Toolkits die Privilegien zu aktualisieren.






58.3 Tabelle 1a

```
CREATE TABLE contact (
  uid SMALLINT NOT NULL AUTO_INCREMENT,
  first_name CHAR(20),
  middle_initial CHAR(1),
  last_name CHAR(30),
  address_1 VARCHAR(100),
  address_2 VARCHAR(100),
  country VARCHAR(30),
  state CHAR(2),
  home_phone VARCHAR(30),
  work_phone VARCHAR(30),
  work_fax VARCHAR(30),
  cell_phone VARCHAR(30),
  email_normal VARCHAR(70),
  email_urgent VARCHAR(70),
  birthday CHAR(11),
  URL VARCHAR(80),
  PRIMARY KEY(uid),
  KEY(last_name,first_name),
  KEY(state),
  KEY(work_phone)
);
```







58.4 Tabelle 1b

```
CREATE TABLE comments (
   uid SMALLINT NOT NULL,
   contact_date DATETIME NOT NULL,
   contact_comment CHAR(60),
   KEY(uid),
   KEY(contact_date),
   KEY(contact_comment)
)
```







58.5 Tabelle 1c

```
mysql CREATE TABLE example (uid SMALLINT NOT NULL  \
AUTO_INCREMENT,f_nameCHAR(20), \
m_initial CHAR(1), \
l_name CHAR(30), \
PRIMARY KEY(uid), \
KEY(l_name,f_name), \
)
```







58.6 Tabelle 1d

```
cat my_create_file | mysql
```




58.7 PHP Listing

```
<HTML>
<?
/* Script Name:    contact.html
Script Purpose: Simple Contact Database Script Version: 1.1
Script Author:  Michael J. Miller Jr. (mke@netcom.com)
*/
/* First let's set some variables. */
$hostname = "localhost";
$password = "grok";
$user = "contact";
/* Now to define functions. */
/* The main screen.  Printed when $state is empty. */
Function Main_Menu (
echo "<H1><CENTER> Welcome To The Contact Database ";
echo "You have the following choices:";
echo "<FORM METHOD = \"POST\" ACTION=\"contact.html\">";
echo "<INPUT CHECKED TYPE=\"radio\" NAME=\"state\" VALUE=\"List\"><B> List
Contacts.<BR>";
echo "<INPUT TYPE=\"radio\" NAME=\"state\" VALUE=\"Create\"><B>Create a new
Contact.<P>";
echo "<INPUT TYPE=\"submit\" VALUE=\"Make Contact\">";
echo "";
);
/* List the first and last name of people in the contact database plus their
phone number.  Make the last name clickable for the full record. */
Function List (
echo "<CENTER><H2>Contact List<P>";
mysql_connect($hosthame,$user,$password);
$result = mysql("ContactDB","SELECT uid, last_name, first_name, phone FROM
contact ORDER BY last_name");
echo "<TABLE BORDER = 10 CELLPADDING = 2>";
$total_rows = mysql_numrows($result); $counter = 0;
echo "<TR><TD><B>Last Name<TD><B>First Name<TD><B>Work
Phone";
while($counter < $total_rows);
$uid = mysql_result($result,$counter,"uid"); echo "<TR><TD>\n";
echo "<A HREF=contact.html?uid=$uid=Print_Contact>";
echo mysql_result($result,$counter,"last_name");
echo "\n";
echo "<TD>\n";
echo mysql_result($result,$counter,"first_name"); echo "<TD>\n";
echo mysql_result($result,$counter,"phone"); echo "\n";
$counter = $counter + 1;
endwhile;
echo "";
);
/* Save updates to database.  Call Print_Contact to list new information
for updated contact entry. */
Function Commit_Update $uid $fn $mi $ln $ph $fx $em (
mysql_connect($hosthame,$user,$password);
```

```
$result = mysql("ContactDB","UPDATE contact SET first_name = '$fn',
middle_initial = '$mi', last_name = '$ln', phone = '$ph', fax = '$fx',
email = '$em' WHERE uid = $uid");
$state = "";
Print_Contact($uid);
);
/* Save new contact information to database.  Call Print_Contact to display
new contact entry. */
Function  Commit_Contact $fn $mi $ln $ph $fx $em (
mysql_connect($hostname,$user,$password);
$result = mysql("ContactDB","SELECT now()"); /* Get the Current Datetime */
/* Note that it would be
better to use a field
of type TIMESTAMP here,
since it would automatically
assign the current date and
time.  I haven't done that
here since I wanted to
demonstrate using functions
in MySQL. */
$now = mysql_result($result,0,"now()");
$result = mysql("ContactDB","INSERT INTO contact
(first_name, middle_initial,last_name, phone, fax, email)
VALUES ('$fn', '$mi','$ln', '$ph', '$fx', '$em')");
$new_uid = mysql_insert_id();
$result = mysql("ContactDB","INSERT INTO comment (uid, contact_date,
contact_comment) VALUES ($new_uid, '$now' , 'Contact Record Created')");
$state = "";
Print_Contact($new_uid);
);
/* This function displays the contact nformation in a form.  It is used
for both the create and update contact options. */
Function Contact_Form $state,$uid (
if($state == "Update");
$state = "Commit_Update";
$result = mysql("ContactDB","SELECT * FROM contact WHERE uid = $uid");
$last_name = mysql_result($result,0,"last_name");
$first_name = mysql_result($result,0,"first_name");
$middle_initial = mysql_result($result,0,"middle_initial");
$phone = mysql_result($result,0,"phone");
$fax = mysql_result($result,0,"fax"); $email =
mysql_result($result,0,"email");
else;
$state = "Commit_Contact";
endif;
echo "<CENTER><H2>Contact Form<P>\n";
echo "<FORM METHOD = \"POST\" ACTION=\"contact.html\">\n";
echo "   <INPUT TYPE = \"HIDDEN\" NAME = \"uid\" VALUE = \"$uid\">\n"; echo
"
<INPUT TYPE = \"HIDDEN\" NAME = \"state\" VALUE = \"$state\">\n";
echo "   First Name: <BR><INPUT TYPE = \"TEXT\" NAME=\"first_name\" \n";
echo "
VALUE = \"$first_name\" MAXLENGTH = 20><BR>\n";
echo "   Middle Initial: <BR><INPUT TYPE = \"TEXT\" NAME=\"middle_initial\"
\n";
echo "   VALUE = \"$middle_initial\" MAXLENGTH = 1 SIZE = 2><BR>\n";
echo "   Last Name: <BR><INPUT TYPE = \"TEXT\" NAME=\"last_name\" \n"; echo
"
```

```
VALUE = \"$last_name\" MAXLENGTH = 30><BR><P>\n";
echo "   Phone: <BR><INPUT TYPE = \"TEXT\" NAME=\"phone\" \n"; echo "
VALUE = \"$phone\" MAXLENGTH = 30><BR>\n";
echo "   Fax: <BR><INPUT TYPE = \"TEXT\" NAME=\"fax\" \n"; echo "
VALUE = \"$fax\" MAXLENGTH = 30><BR>\n";
echo "   Email: <BR><INPUT TYPE = \"TEXT\" NAME=\"email\" \n"; echo "
VALUE = \"$email\" MAXLENGTH = 70 SIZE = 40><P>\n";
echo "   <CENTER><INPUT TYPE=\"submit\" VALUE=\"Submit
Contact\">\n";
echo "\n";
);
/* Add a comment to the database.  Call Print_Contact to display changes. */
Function Add_Comment $uid $contact_comment (
mysql_connect($hosthame,$user,$password);
$result = mysql("ContactDB","SELECT now()"); /* Get the Current Datetime */
$now = mysql_result($result,0,"now()");
$result = mysql("ContactDB","INSERT INTO comment (uid, contact_date,
contact_comment) VALUES ($uid, '$now', '$contact_comment')");
Print_Contact($uid);
);
/* Print address and comments for a contact.  Also give option to update
or add comments. */
Function Print_Contact $uid (
mysql_connect($hosthame,$user,$password);
$result = mysql("ContactDB","SELECT * FROM contact WHERE uid = $uid");
$name = mysql_result($result,0,"first_name");
$name = $name + " " + mysql_result($result,0,"middle_initial");
$name = $name + " " + mysql_result($result,0,"last_name");
echo "<TITLE>Contact Information For $name"; echo "";
echo "<HTML>";
echo "<CENTER><H2>Contact Information For $name";
echo "<TABLE BORDER = 10 CELLPADDING = 2>";
echo "<TR><TD>Name";
echo "<TD>$name";
$result = mysql("ContactDB","SELECT * FROM contact WHERE uid = $uid");
echo "<TR><TD>Phone";
echo "<TD>";
echo mysql_result($result,0,"phone"); echo "\n";
echo "<TR><TD>Fax";
echo "<TD>";
echo mysql_result($result,0,"fax");
echo "\n";
$email = mysql_result($result,0,"email");
echo "<TR><TD>Email Address";
echo "<TD><A HREF=\"MAILTO:";
echo "$email \"> $email";
echo "\n";
echo "";
echo "<FORM METHOD = \"POST\" ACTION=\"contact.html\">";
echo "   <INPUT TYPE = \"HIDDEN\" NAME = \"uid\" VALUE = \"$uid\">\n"; echo
"
<INPUT TYPE = \"HIDDEN\" NAME = \"state\" VALUE = \"Update\">\n";
echo "<CENTER><INPUT TYPE=\"submit\" VALUE=\"Update Contact
Information\">";
echo "<P><HR WIDTH=\"100%\">";
echo "<CENTER><H2>Comments For This Contact";
echo "";
$result = mysql("ContactDB","SELECT * FROM comment WHERE uid = $uid ORDER BY
```

```
contact_date");
$total_rows = mysql_numrows($result); $counter = 0;
echo "<TABLE BORDER = 10 CELLPADDING = 2>"; echo
"<TR><TD>Date<TD>Comment";
while($counter < $total_rows);
echo "<TR><TD>";
echo mysql_result($result,$counter,"contact_date"); echo "<TD>";
echo mysql_result($result,$counter,"contact_comment"); echo "";
$counter = $counter + 1;
endwhile;
echo "";
echo "<P>Add new comment below.";
echo "<FORM METHOD = \"POST\" ACTION=\"contact.html\">";
echo "<P><INPUT TYPE = \"TEXT\" NAME = \"contact_comment\" ";
echo "MAXLENGTH = 60 SIZE = 60><BR>";
echo "<INPUT TYPE = \"HIDDEN\" NAME = \"uid\" VALUE = \"$uid\"><P>";
echo "<INPUT TYPE = \"HIDDEN\" NAME = \"state\" VALUE =
\"Add_Comment\"><P>";
echo "<CENTER><INPUT TYPE=\"submit\" VALUE = \"Submit Comment\">";
echo "<P>";
echo "<FORM METHOD = \"POST\" ACTION=\"contact.html\">";
echo "<CENTER><INPUT TYPE=\"submit\" VALUE=\"Return To Main
Menu\">";
echo "\n";
);
/* The main loop.  Call functions based on the value of $state, which
gets set via a hidden INPUT TYPE. */
switch($state) {
case "";
Main_Menu();
break;
case "List";
List();
break;
case "Create";
Contact_Form($state, $uid);
break;
case "Update";
Contact_Form($state, $uid);
break;
case "Commit_Update";
Commit_Update($uid,$first_name,$middle_initial,$last_name,$phone,$fax,$ema
il);
break;
case "Commit_Contact";
Commit_Contact($first_name,$middle_initial,$last_name,$phone,$fax,$email);
break;
case "Add_Comment";
Add_Comment($uid,$contact_comment);
break;
case "Print_Contact";
Print_Contact($uid);
endswitch;
>
```




59. LAMP (Linux Apache MySQL PHP) und CHROOT()

Viele Distributionen bieten bereits von Hause aus eine LAMP Installation an. Allen gemeinsam ist, daß man diese niemals im Internet als Server verwenden sollte. Der Apache Server alleine ist in der Minimalkonfiguration sehr sicher und sozusagen **bullet proof**. Problematisch wird es bei CGI-BIN´s und Datenbank-Modulen. Im Firewall Handbuch http://www.little-idiot.de/firewall/ sind Möglichkeiten beschrieben, wie man z.B. PERL CGI_BIN´s so programmieren kann, daß diese recht sicher sind. PERL bietet im sog. Taint Modus eine hohe Sicherheit, sodaß Angreifer keine Befehle in das System einschleusen können. Beim Einsatz von PHP 3/4 muß ich jedoch sagen, daß es kaum Sicherungsmöglichkeiten gibt. Der einzige Ausweg ist eine CHROOT() Umgebung, wo quasi das oberste Verzeichnis, welches ein User erreichen kann, z.B. **/home/www/** ist. Auf Verzeichnisse anderer User oder sogar auf das Verzeichnis **/etc/** hat ein Angreifer dann keine Zugriffsmöglichkeit mehr.

Man kann so eine quasi vom Rest des Servers gekapselte Benutzerumgebung schaffen, aus der auf gewiefte Angreifer nicht mehr entkommen können. Falls in dieser Umgebung dann der Apache und MySQL Server läuft, dann kann ein Angreifer zwar den Betrieb stören, jedoch bleiben die Log-Dateien in **/var/log/** für ihn unerreichbar. Er läuft in Gefahr, entdeckt zu werden und verabschiedet sich schon deswegen von bösen Absichten.

Leider enthält keines der Handbücher über PHP, PERL oder den Apache WWW-Server Lösungsmöglichkeiten. Im Firewall Handbuch ist ausführlich ein Skript beschrieben, wie man eine CHROOT() Umgebung aufbaut. Hierzu gibt es jedoch einiges bei der Installation zu beachten:

1. Die Serverdämonen Apache, MySQL sowie die Libraries für PHP müssen völlig autak z.B. im Verzeichnis **/home/www** installiert laufen können. Ein Zugriff auf /lib/, /usr/local, /usr/lib, u.s.w. wiederspricht den Erfordernissen einer CHROOT() Umgebung.
2. Alle Quellcodes, Bibliotheken u.s.w. müssen sich vor der Kompilation in einer CHROOT() Umgebung befinden.
3. Wenn es gelingt, alle Programme als normaler User zu kompilieren, dann haben wir fast sichergestellt, daß keine Routinen außerhalb der CHROOT() Umgebung mehr angesprochen werden müssen.
4. Statisches Kompilieren verhindet, daß Teile von Libraries nachgeladen werden müssen. Das kostet zwar mehr RAM, ist aber die Voraussetzung dafür, daß die Serverdämonen in der CHROOT() Umgebung autak laufen.
5. Mit Hilfe von **strace** oder **ptrace** kann man als User sehen, ob eventuell noch Bibliotheken o.ä. außerhalb der CHROOT() Umgebung angesprochen werden.
6. Danach erst, wenn alles korrekt läuft, kann man das CHROOT() Skript verwenden, um die Dämonen einzuschließen.

Unser größtes Problem ist nun, Den Apache-Server, den MySQL Server, PHP und die wichtigsten Routinen unterhalb des Verzeichnisses **/home/www/** ohne irgendwelche Superuser-Rechte zu

installieren.



59.1 Apache WWW-Server Installation unter /home/www/

Wir testen nun zuerst die Installation eines Apache WWW-Servers als Vorbereitung. Loggen Sie sich als Superuser oder Administrator ein und legen Sie sich bitte einen User **www** an. Loggen Sie sich aus und als User **www** wieder ein, und kopieren Sie die Apache Quellcodes in das Homeverzeichnis des Users **www**. Nur wenn Sie Apache als normaler User **www** installieren und starten, können Sie sicher sein, daß ein Cracker im schlimmsten Falle auch nur diese Rechte erhält. Die Installation des Apache Servers als User **www** und der Start des Apache Server ist die Grundvoraussetzung dafür , daß der Apache Server auch in einer CHROO() Umgebung läuft. Wer Apache als Systemadministrator installiert, riskiert eine Fehlkonfiguration und eventuelle Sicherheitslücken im System. Hier nun die Ausgangsposition. Gehen Sie nun Schrittweise vor:

```
www@tunix:~ > ls
apache_1.3.9.tar.gz
www@tunix:~ > whoami
www
www@tunix:~ > tar -xzvf apache_1.3.9.tar.gz
...
...
www@tunix:~ > cd apache_1.3.9
www@tunix:~/apache_1.3.9 >
```

Kopieren Sie nun das Hilfsskript mit CUT - und PASTE in dieses Verzeichnis (Markieren mit der linken Maustaste und pasten mit der mittleren oder beiden Maustasten, mit STRG-d beenden Sie den Vorgang). Passen Sie eventuell in dem Skript noch die Versionsnummer des Apache Servers an:

```
#!/bin/sh
# Apache Konfigurations-Datei apache.conf

./configure \
        --prefix=/home/www/usr/local/apache-1.3.9 \
        --mandir=/home/www/usr/local/apache-1.3.9/man \
        --logfiledir=/home/www/usr/local/apache-1.3.9/logs \
        --sysconfdir=/home/www/usr/local/apache-1.3.9/conf \
        --datadir=/home/www/usr/local/apache-1.3.9/htdocs \
        --enable-module=info \
        $*
```

Wir speichern nun diess Skrit mit Hilfe von Drag - und Drop in ein Verzeichnis. Hierzu markieren wird das Skript im Browser schwarz, öffenen eine zweite Shell und geben dann folgendes ein. Mit der mittleren Maustaste oder den beiden äußeren Maustasten kann man dann **pasten**:

```
www@tunix:~/apache_1.3.9 > cat >> hilfsskript.sh
#!/bin/sh
# Apache Konfigurations-Datei apache.conf

./configure \
        --prefix=/home/www/usr/local/apache-1.3.9 \
        --mandir=/home/www/usr/local/apache/man \
        --logfiledir=/home/www/usr/local/apache/logs \
        --sysconfdir=/home/www/usr/local/apache/conf \
        --datadir=/home/www/usr/local/apache/htdocs \
```

```
        --enable-module=info \
        $*
```

Sie haben nun eine Datei **hilfskript.sh** angelegt, die noch ausführbar gemacht werden muß:

```
www@tunix:~/apache_1.3.9 > chmod u+x hilfsskript.sh
www@tunix:~/apache_1.3.9 >
```

Das Skript ist nun ausführbar und kann nach einer eventuellen Anpassung der Versionnummer des Apache-Servers (1.3.9) gestartet werden:

```
www@tunix:~/apache_1.3.9 > hilfsskript.sh
Configuring for Apache, Version 1.3.9
 + using installation path layout: Apache (config.layout)
Creating Makefile
Creating Configuration.apaci in src
Creating Makefile in src
 + configured for Linux platform
 + setting C compiler to gcc
 + setting C pre-processor to gcc -E
 + checking for system header files
 + adding selected modules
 + checking sizeof various data types
 + doing sanity check on compiler and options
Creating Makefile in src/support
Creating Makefile in src/regex
Creating Makefile in src/os/unix
Creating Makefile in src/ap
Creating Makefile in src/main
Creating Makefile in src/lib/expat-lite
Creating Makefile in src/modules/standard
www@tunix:~/apache_1.3.9 >
```

Nun können Sie den Apache Server kompilieren und installieren:

```
www@tunix:~/apache_1.3.9 > make all; make install
...
...
./src/helpers/install.sh -c -m 644 ./conf/mime.types
/home/www/conf/mime.types.default
./src/helpers/install.sh -c -m 644 ./conf/mime.types
/home/www/conf/mime.types
./src/helpers/install.sh -c -m 644 ./conf/magic /home/www/conf/magic.default
./src/helpers/install.sh -c -m 644 ./conf/magic /home/www/conf/magic
<=== [config]
make[1]: Leaving directory /home/www/apache_1.3.9'
+--------------------------------------------------------+
| You now have successfully built and installed the      |
| Apache 1.3 HTTP server. To verify that Apache actually |
| works correctly you now should first check the         |
| (initially created or preserved) configuration files   |
|                                                        |
|   /home/www/conf/httpd.conf
|                                                        |
| and then you should be able to immediately fire up     |
| Apache the first time by running:                      |
|                                                        |
```

```
|   /home/www/usr/local/apache-1.3.9/bin/apachectl start
|                                                      |
| Thanks for using Apache.       The Apache Group      |
|                                http://www.apache.org/  |
+-------------------------------------------------------+
www@tunix:~/apache_1.3.9 >
```

Der Apache Server ist nun unter einem Useraccount installiert und muß nun vor dem Start konfiguriert werden. Das Konfigurationsskript befindet sich nun im Verzeichnis **conf**:

```
www@tunix:~/apache_1.3.9 > ls
ABOUT_APACHE      Makefile.tmpl     conf              icons
Announcement      README            config.layout     logs
INSTALL           README.NT         config.status     src
KEYS              README.configure  configure
LICENSE           WARNING-NT.TXT    hilfsskript.sh
Makefile          cgi-bin           htdocs
www@tunix:~/apache_1.3.9 > cd conf
www@tunix:~/apache_1.3.9/conf > ls
access.conf-dist           magic
access.conf-dist-win       mime.types
highperformance.conf-dist  srm.conf-dist
httpd.conf-dist            srm.conf-dist-win
httpd.conf-dist-win
www@tunix:~/apache_1.3.9/conf >
```

Die Dateien sind noch mit der Endung **-dist** benannt. Es gibt Windows Konfigurationsdateien und UNIX Konfigurationsdateien. Unter NT und OS/2 können Sie, sofern Sie den GCC oder EGCS Kompiler und die GNU Tools installiert haben, den Apache Server nach demselben Schema installieren. Die Konfigurationsdatei des Apache ist nun **/home/www/usr/local/apache-1.3.9/conf/httpd.conf**. Wir kopieren uns die UNIX Konfigurationsdateien:

```
www@tunix:~/apache_1.3.9/conf > cd ; cd conf; ls
ww@tunix:~/conf > ls
access.conf          magic               srm.conf
access.conf.default  magic.default       srm.conf.default
httpd.conf           mime.types
httpd.conf.default   mime.types.default
www@tunix:~/conf >
```

Im Grunde ist dies die Grundinstallation des Apache, nur halt um **/home/www** in einen Ast des Verzeichnisbaumes verschoben. Damit das Apache Binary auch die Verzeichnisse kennenlernt, haben wir die Konfigurations-Optionen entsprechend angepaßt.

Wir können nun z.B. mit dem Editor **joe** noch einige Anpassungen vornehmen. Mit STRG-k x können Sie die Datei speichern. Der Apache Server ist nun auf Port 8080 konfiguriert, damit es keine Kollision mit bestehenden Server gibt. Er läuft mit den User-Rechten **nobody** und **nogroup**. Falls dieser User oder diese Gruppe nicht vorhanden ist, benutzen Sie vorerst **www** und **users**.

Um den Apache Server zu starten, geben Sie folgendes ein: (sie sehen nur die Enden der Verzeichnisse am Prompt...)

```
www@tunix:~/conf > cd
www@tunix:~ > cd usr/local/apache-1.3.9/bin/
www@tunix:~/usr/local/apache-1.3.9/bin > httpd
www@tunix:~/usr/local/apache-1.3.9/bin >
```

Sie haben nun den Apache Server als einfacher User auf dem Port 8080 gestartet. Geben Sie nun in Ihren Browser die IP-Nummer Ihres WWW-Server mit dem Anhängsel **:8080** an, z.B. **http://10.0.0.5:8080**. Sie sehen nun die

Standardmeldung eines erfolgreich installierten Apache Servers, jedoch ohne PHP Module.








59.2 Nachträgliche Installation von PHP 3 (beta)

Die nachträgliche Installation von PHP 3 ist mit der Neukompilation des Apache verbunden. Hierzu muß auch unser Konfigurationsskript **hilfsskriptphp** angepaßt werden:

```
#!/bin/sh
# Apache Konfigurations-Datei apache.conf

./configure \
        --prefix=/home/www/usr/local/apache-1.3.9 \
        --mandir=/home/www/usr/local/apache-1.3.9/man \
        --logfiledir=/home/www/usr/local/apache-1.3.9/logs \
        --sysconfdir=/home/www/usr/local/apache-1.3.9/conf \
        --datadir=/home/www/usr/local/apache-1.3.9/htdocs \
        --enable-module=info \
        --activate-module=src/modules/php3/libphp3.a \
        --enable-shared=php3
        $*
```

Im Grunde sind nur die letzen 3 Zeilen hinzugefügt worden.

Problematisch wird es aber mit PHP 3 und den vielen Libraries. Bei der Kompilation verwendet PHP viele installierte Routinen aus den Verzeichnissen **/usr/local, /usr/lib/, /lib** u.s.w. Wir müssen also den mühseligen Weg gehen, die Quellcodes von MySQL, der GD-Lib, der ZLIB, von PHP 3 u.s.w. aus dem Internet zu laden, und unter **/home/www/** zu installieren. PHP 3 kann erste dann kompiliert werden, wenn alle Libraries erkannt und im Konfigurationsfile eingebunden sind. Dann können wird das PHP Modul bzw. die PHP Libraries für den Apache Server erzeugen und danach erst läßt sich der Apache Server kompilieren. Alles klar ? Wohl kaum ! Daher also Schritt für Schritt:







59.3 Apache und PHP 3

Nun werden wir Schritt für Schritt also zuerst einmal PHP 3 in der Grundkonfiguration in den Apache Server hineinkompilieren:

Zuerst aber müssen wir die PHP - Libraries erzeugen. Laden Sie sich vom Server http://www.php3.net die Quellen zu PHP3. Diese heißen in unsererm Beispiel **php-3.0.12.tar.gz**. Kopieren Sie diese neben das Apache Archiv in **/home/www** und entpacken Sie die Quellcodes mit **tar -xzvf php-3.0.12.tar.gz**. Es wird ein Verzeichnis **php-3.0.12** angelegt.

```
www@tunix:~ > cd
www@tunix:~ > cd php-3.0.12
www@tunix:~/php-3.0.12 > ./configure --prefix=/home/www/www --with-mysql \
        --with-apache=../apache_1.3.9 --with-axps
...
...
creating libphp3.module
creating extra/gd/bdf2gdfont
creating regex/Makefile
creating pcrelib/Makefile
creating dbase/Makefile
creating config.h

www@tunix:~/php-3.0.12 > make ; make install
www@tunix:~/php-3.0.12 >
```

Sie haben nun PHP3 mit allen erdenklichen Modulen für verschiedenste SQL Server, Grafikbibliotheken und Funktionen für den Zugriff über Sockets auf andere Server u.s.w. kompiliert. Damit jedoch MySQL auch korrekt läuft, müssen die MySQL Bibliotheken in **/usr/lib/mysql/** installiert sein.

Falls Sie gedenken, den Apache Server in einer CHROOT() Umgebung zu installieren, müssen Sie entweder alles statisch kompilieren, oder die Systemlibraries in die CHROOT() Umgebung kopieren. Lesen Sie hierzu das Firewall Handbuch: http://www.little-idiot.de/firewall/.

Wechseln Sie nun in das Verzeichnis des Apache-Servers, und führen Sie folgende Kommandos aus:

```
www@tunix:~/php-3.0.12 > cd
www@tunix:~ > cd apache_1.3.9
root@tunix:/home/www/apache_1.3.9 > ./configure
--prefix=/home/www/usr/local/apache-1.3.9
--activate-module=src/modules/php3/libphp3.a
--datadir=/home/www/usr/local/apache-1.3.9/htdocs
--enable-module=info
```

Mit **make; make install** können Sie dann den Apache WWW-Server neu kompilieren. Damit dieser auch PHP 3 Skripte korrekt interpretiert, müssen Sie (entgegen den Anleitungen der Apache Dokumentation) in der Datei **srm.conf** zwei Zeilen hinzufügen:

```
AddType application/x-httpd-php3 .php3
```

```
AddType application/x-httpd-php3-source .phps
```

Sie sollten auch in der Datei **/home/www/usr/local/apache-1.3.9/conf/httpd.conf** den Port, die Log - Verzeichnisse und evtl. das **DocumentRoot** Verzeichnis (/home/www/usr/local/apache-1.3.9/htdocs/) anpassen.

Nun können Sie mit **/home/www/usr/local/apache-1.3.9/bin/apachectl start** den Apache Server neu kompilieren und installieren. Dieser Apache WWW-Server wird nun viele PHP 3 Skripte korrekt ausführen.

Einige Dinge werden noch nicht funktionieren. Das sind die Ansteuerung von MySQL und die tollen Grafikroutinen von Thomas Boutell, mit denen man in Echtzeit Schrift auf Buttons schreiben kann....

Wenden wir uns nun also den vielen Routinen für PHP3 zu, und versuchen, diese in der Umgebung **/home/www/** zu installieren und zu kompilieren...

Vergessen Sie alle Distributionen und RPM, DEB oder PKG Dateien. Diese Softwarepakete können unsere Ansprüche ohnehin nicht alle erfüllen. Wenn Sie nach dieser Anleitung vorgehen, dann können Sie sogar LINUX 1.2.13 nach dem Stand von vor 3 Jahren verwenden. Vergessen Sie also einmal den Versions-Hype der LINUX Distributoren - es ist absoluter Blödsinn, sich stets eine aktuelle LINUX Distribution zu kaufen, wenn man mal wieder ein kleines Feature mehr benötigt, wie z.B. PHP3 oder PHP4. Ich persönlich arbeite sogar noch mit einer Version 1.2.13. Der Server läuft seit ca. 2 Jahren ohne Unterbrechung. Inzwischen läuft auch das aktuelle LAMP System mit MySQL 3.23 und Apache 1.3.9 darauf.

Bevor wir nun den Apache Server bauen, zunächst einige Grundlagen:

Ein Makefile ist dafür verantwortlich, das komplette System zu erstellen und zu installieren. Wenn man standardmäßig **make all** angibt, dann läuft man in Gefahr, eine alte Installation zu überschreiben, oder daß das sogenannte **DocumentRoot**, also das Homeverzeichnis des Apache WWW-Servers in einem Verzeichnis installiert wird, an welches man mit Hilfe des FTP Servers zum Upload von HTML, PHP3, CGI-BIN´s nicht herankommt. Daher gibt man beim Kompilieren einige Optionen an, die dieses Problem lösen:

```
#!/bin/sh
# Apache Konfigurations-Datei apache.conf

./configure \
        --prefix=/usr/local/apache-1.3.9 \
        --mandir=/usr/local/man \
        --logfiledir=/home/www/logs \
        --sysconfdir=/home/www/conf \
        --datadir=/home/www \
        --enable-module=info \
        $*
```

Dieses Skript starten das Konfigurations - Skript **.configure** mit den Optionen, die schließlich veranlassen, daß einige Teile im HTTPD so verändert werden, daß der Apache Server stets das Verzeichnis **/home/www** als Verzeichnis für die HTML Dateien betrachtet. Zu einem späteren Zeitpunkt können Sie dann beliebig viele, virtuelle WWW-Server hinzufügen. Sie können auch mehrere Apache Versionen parallel installieren, wobei z.B. die neue Version auf Port 81 läuft. Mit Hilfe der Option **Redirect www.domain.de www.domain.de:81** in dem Konfigurationsfile Ihres Apache Servers auf Port 80 können Sie dann veranlassen, daß diese Domain dann auf den neuen Apache Server umgeleitet wird. So können Sie auf einem einzigen Server zwei oder mehrere Versionen des Apache WWW-Servers für utnerschiedliche Kunden installieren. Versuchen Sie das einmal mit Windwows NT !

Der Apache WWW-Server benötigt nur wenige Konfigurationsanweisungen da bisher keine zusätzlichen Module eingebunden werden. Lediglich das mitgelieferte Info-Modul wird aktiviert, da dies die Fehlersuche in einem LAMP-Systems erleichtert. Apaches Log-Dateien werden in /home/www/logs angelegt, seine Konfigurationsdateien in /home/www/conf und die die Daten der Beispielseiten werden in /home/www/htdocs gespeichert. Dadurch wird

eine einfache Datensicherung gewährleistet. Legen Sie diese Verzeichnisse so mit **mkdir** an, oder ändern Sie diese so ab, daß Sie mit Ihren Vorstellugnen von Userverzeichnissen übereinstimmt.

Das Makefile selber sieht folgendermaßen aus. Sie können aber auch das originale Makefile verwenden, welches nach Ausführung von **.configure** oder obigem Skript erstellt wurde:






60. Danksagung

Rasmus Lerdorf und allen Entwicklern von PHP: http://www.php.net/credits.php3 den Entwicklern von MySQL, MSQL ..... (to be continued)